MacroQC : An Electronic Structure Theory Software for Large-Scale Applications

User's Manual

Uğur Bozkaya,[a] Betül Ermiş,[a] Yavuz Alagöz,[a] Aslı Ünal,[a] and Ali Kaan Uyar[a]

[a] *Department of Chemistry, Hacettepe University, Ankara 06800, Turkey*

MacroQC Version: 2.0
Created on: September 6, 2026
Git Version: 51f31d7
Report bugs to: macroqcprog@gmail.com

Contents

1 Overview **8**

1.1 About 8

1.2 Capabilities 10

2 Citing `MacroQC` **11**

2.1 Overall `MacroQC` Package 11

2.2 Orbital-Optimized Post-Hartree-Fock Methods 11

2.2.1 Orbital-Optimized Second-Order Perturbation Theory (OMP2) . . . 11

2.2.2 Orbital-Optimized Third-Order Perturbation Theory (OMP3) and Orbital-Optimized MP2.5 (OMP2.5) 12

2.2.3 Orbital-Optimized Linearized Coupled-Cluster Doubles Method (OLCCD) 12

2.2.4 Orbital-Optimized Coupled-Cluster Doubles Method (OCCD) 13

2.2.5 Orbital-Optimized Coupled-Cluster Doubles Method with Perturbative Triples [OCCD(T) and OCCD(T)$_\Lambda$] 13

2.3 Perturbation Theory (PT) 13

2.4 Coupled-Cluster (CC) 13

2.5 Second-Order Quasidegenerate Perturbation Theory (QDPT2) 14

2.6 Extended Koopmans' Theorem (EKT) 14

2.7 Molint Framework 15

2.8 Dipole Moments and One-Electron Properties 15

2.9 Anharmonic Force Field and IR Spectra 15

2.10 Linear-Scaling Systematic Molecular Fragmentation 15

2.11 Equation-of-Motion Coupled-Cluster 15

3 Download and Installation **16**

3.1 Install with Conda (Linux and Mac OS X) 16

3.2 Install with Self-Extracting Package (Linux and Mac OS X) 16

3.3 Manual Installation from Zip Archive (Linux and Mac OS X) 16

4 Testing `MacroQC` **17**

5 Running `MacroQC` **18**

5.1 Setting Environmental Variables 18

5.2 Running MacroQC in Serial Mode 19
5.3 Running MacroQC in Parallel Mode Using OpenMP 19
5.4 Running Calculations Across Nodes Using MPI 19

6 Basic Input Format **21**
6.1 Input Examples 24
6.2 Ghost Atoms 24
6.3 Z-Matrix Input 25
6.4 External Charges 26
6.5 Multiple Jobs 27
6.6 Global Options 29
6.7 Default Values for the Frozen Core Approximation 35

7 Basis Sets **36**
7.1 Overview 36
7.2 Built-In Basis Sets 36
7.3 Assigning Basis Sets for Individual Atoms 44
7.4 User-Defined Basis Sets 46
7.5 Integral Library Options 47

8 SCF : Self-Consistent Field **48**
8.1 Overview 48
8.2 ROHF 49
8.2.1 The ROHF gradient 49
8.3 Quasi-Restricted Orbitals 50
8.4 Input Examples 50
8.5 SCF Module Options 52

9 OPT : Geometry Optimization **61**
9.1 Overview 61
9.2 Relaxed scans 61
9.3 Freezing string method 62
9.4 Growing string method 64
9.5 Single-ended growing string 66
9.6 Which of the three 69

9.7 Nudged elastic band . . . 70
9.8 IRC Overview . . . 71
9.9 Input Examples . . . 72
9.9.1 Constrained optimization . . . 76
9.10 OPT Module Options . . . 79
9.11 SCAN Module Options . . . 89
9.12 FSM Module Options . . . 90
9.13 GSM Module Options . . . 94
9.14 NEB Module Options . . . 97
9.15 IRC Module Options . . . 99

10 FREQ: Vibrational Frequencies 103
10.1 Overview . . . 103
10.2 Input Examples . . . 104
10.3 FREQ Module Options . . . 107

11 DFOCC : Orbital-Optimized Coupled-Cluster and Moller–Plesset Perturbation Theories 111
11.1 Theory . . . 111
11.2 Conventional (Non-OO) Coupled-Cluster and Moller-Plesset Perturbation Theories . . . 113
11.3 Input Examples . . . 114
11.4 DFOCC Module Options . . . 117

12 LPNOCC : Local Pair-Natural Orbital Coupled-Cluster Methods 130
12.1 Overview . . . 130
12.2 Accuracy of the LPNO Approximation . . . 130
12.3 PNO Module Options . . . 132
12.4 LPNOCC Module Options . . . 135
12.5 LPNO-MP2 and the Internal Fock Block . . . 137
12.6 Open-Shell LPNO-MP2 . . . 137
12.7 Open-Shell LPNO-LCCD and LPNO-CCD . . . 139
12.8 Open-Shell LPNO-CCSD(T) . . . 142
12.9 Behaviour with Truncation . . . 144
12.10Input Examples . . . 144

13 CIM: Cluster-in-Molecule Methods **145**
13.1 Overview . . . 145
13.2 Localization and the CIM Energy Decomposition . . . 146
13.3 Single-Environment CIM (SECIM) . . . 147
13.3.1 Central and Environmental LMOs . . . 147
13.3.2 Collapsing Embedded Domains . . . 147
13.3.3 AO Domains and Buffer Atoms . . . 147
13.3.4 Projected Orbitals and Virtual Space . . . 148
13.4 Improved CIM (ICIM) . . . 148
13.4.1 AO Domain of an MO Domain . . . 148
13.4.2 Approximate PAOs and the Composite Domain . . . 149
13.4.3 Virtual Space . . . 149
13.5 Semicanonicalization and Assembling the Total Energy . . . 149
13.6 CIM Module Options . . . 150
13.7 Input Examples . . . 152

14 DEC: Divide-Expand-Consolidate Methods **153**
14.1 Overview . . . 153
14.2 Atomic Sites and Orbital Extents . . . 154
14.3 Fragment and Pair Energies . . . 154
14.4 DEC Module Options . . . 154
14.5 Input Example . . . 156

15 ML: Machine-Learning Models **156**
15.1 Overview . . . 156
15.2 Molecular Descriptors . . . 157
15.3 Models . . . 158
15.4 Training . . . 159
15.5 ML Module Options . . . 159
15.6 Input Example . . . 161

16 QDPT: Quasidegenerate Perturbation Theory **162**
16.1 Input Examples . . . 162
16.2 QDPT Module Options . . . 164

17 DFT: Kohn–Sham Density Functional Theory **169**
17.1 Overview . . . 169
17.2 Functionals . . . 169
17.3 The Integration Grid . . . 170
17.3.1 NUMQUAD Module Options . . . 170
17.4 Chain-of-Spheres Exchange . . . 171
17.4.1 COSX Options (SCF block) . . . 172
17.5 Empirical Dispersion . . . 172

18 CIS and TDA/TD-DFT: Excited States **174**
18.1 Overview . . . 174
18.2 Algorithms . . . 174
18.3 CIS Module Options . . . 175
18.4 TDA and TD-DFT . . . 176

19 EOMEE : Equation-of-Motion Coupled-Cluster **178**
19.1 Input Examples . . . 178
19.2 EOMEE Module Options . . . 178

20 Fragment: Molecular Fragmentation Approaches **181**
20.1 FRAGMENT Module Options . . . 181
20.2 Linear-Scaling Systematic Systematic Molecular Fragmentation (LSSMF) . . . 183
20.2.1 Input Examples . . . 183
20.2.2 LSSMF Options . . . 200
20.3 Generalized N-Body Fragmentation . . . 211
20.3.1 Input Examples . . . 211
20.3.2 N-BODY Options . . . 213
20.4 Molecular Fractionation with Conjugate Caps (MFCC) . . . 214
20.4.1 Input Examples . . . 214
20.4.2 MFCC Options . . . 217

21 AIMD Simulations: Ab-Initio Molecular Dynamics **218**
21.1 Overview . . . 218
21.2 Statistical Ensembles . . . 219
21.3 Thermostats . . . 219

21.4 Rigid Bonds . . . 220
21.5 Wavefunction Extrapolation . . . 221
21.6 Multilayer AIMD . . . 221
21.7 The Isothermal-Isobaric Ensemble (NPT) . . . 222
21.8 Output Files . . . 223
21.9 Restarting . . . 224
21.10 AIMD Module Options . . . 224
21.11 Input Examples . . . 230

22 Solvation **233**
22.1 Overview . . . 233
22.2 The Cavity . . . 234
22.2.1 GEPOL Module Options . . . 234
22.3 COSMO . . . 235
22.3.1 COSMO Module Options . . . 236
22.4 PCM: C-PCM and IEF-PCM . . . 237
22.4.1 PCM Module Options . . . 238
22.5 SMD . . . 241
22.5.1 SMD Module Options . . . 241
22.6 CMIRS . . . 242
22.6.1 CMIRS Module Options . . . 243
22.7 COSMO-RS . . . 245
22.7.1 COSMO-RS Module Options . . . 245
22.8 Class-IV Charges . . . 246
22.8.1 Options . . . 246
22.9 The Generalized-Born Model . . . 246
22.9.1 GB Module Options . . . 247
22.10 SM12 . . . 249
22.11 VASPsol . . . 250
22.11.1 VASPSOL Module Options . . . 251
22.12 What Is Available for Each Model . . . 253
22.13 Available Solvents . . . 254

1 Overview

1.1 About

MacroQC is an electronic structure theory software for high-accuracy computations and large-scale chemical applications. The MacroQC software is developed and maintained by Prof. Uğur Bozkaya and his research group at the Department of Chemistry, Hacettepe University, Ankara, Turkey.

The MacroQC software avoids the usage of the conventional four-index electron repulsion integrals (ERIs). Instead, it uses the density-fitting (DF) technique [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19] for all methods included. MacroQC package features embracingly rich perturbation theory and coupled-cluster theory codes, as well as their orbital-optimized variants, for energy and analytic gradient computations. The attractive feature of the MacroQC software is its molecular fragment module, which enables applications of high-accuracy quantum chemical methods to large-scale chemical systems, such as biomolecules and polymers. The developed linear-scaling techniques based on molecular fragmentation approaches enable high-accuracy computations for large molecules.

The MacroQC package originated from the modules and libraries developed by Prof. Uğur Bozkaya. The starting point of MacroQC is the development of the Tensors library, which enables numerous tensor operations. The Tensors library is developed and maintained by Bozkaya since 2013. All ab initio methods and other libraries in the MacroQC package are taking advantage of the Tensors library.

The second important milestone of MacroQC is the development of the DFOCC module, which includes many perturbation theories and coupled-cluster codes. The DFOCC module is developed and maintained by Bozkaya since 2013. Recently, Asli Ünal and Yavuz Alagöz have made contributions to the latest version of the DFOCC module. An older version of the DFOCC module is also available in the Psi4 program package (https://psicode.org).

The third important milestone of MacroQC is the development of basis set and molecular integrals libraries (Molint framework) by Uğur Bozkaya in 2019-2021 [20]. The Molint library provides molecular integrals and their first derivatives, over contracted Gaussian functions, for the density-fitted methods.

Development of the MacroQC software officially started in 2019. The skeleton of the software is built by Bozkaya, Ermiş, and Alagöz. The first module of MacroQC, developed by Bozkaya and Alagöz, is the SCF module. Later, Bozkaya's DFOCC module is integrated with MacroQC by Alagöz. Further, the QDPT module of Bozkaya, which includes Quasidegenerate Perturbation Theory and FCI codes, was incorporated into the MacroQC software by Bozkaya. Bozkaya also developed EKT, FNO, and PNO codes for MacroQC.

Yavuz Alagöz and Betül Ermiş made important contributions to the MacroQC package. They are involved in the development of many core libraries. Yavuz Alagöz contributed to the SCF code and unrestricted (T) code. Alagöz developed the I/O, Timer, Memory, Master, and Options libraries of MacroQC with contributions from Ermiş and Bozkaya. Further, Alagöz converted Bozkaya's DIIS codes into an efficient library.

Betül Ermiş made contributions to the Molecule library of MacroQC, which is responsible for parsing molecular geometry and other related issues. She also developed a geometry optimization module with Prof. Bozkaya. Ermiş significantly extends the applicability of Bozkaya's initial Infrared code and converts it into an efficient API adding many new features to it. Betül Ermiş made substantial contributions to Bozkaya's fragment module and converted it into an efficient API adding many new features.

Asli Ünal made contributions to the MacroQC package. Unrestricted CCSD and EOM-CCSD codes were developed by Ünal and Bozkaya. She keeps work on developing new excited-state methods for the MacroQC software. Finally, Ali Kaan Uyar has joined the MacroQC team. He is the primary developer of our web page, and he is developing local methods for MacroQC, which will be available in future releases.



1.2 Capabilities

Energy and analytic gradient methods that available in the `MacroQC` software are given in Table 1.

Table 1: Capabilities of the `MacroQC` software.

Method	**Energy**	**Gradient**[a]
SCF	RHF | UHF | ROHF	RHF | UHF
MP2	RHF | UHF | ROHF	RHF | UHF
MP2.5	RHF | UHF	RHF | UHF
MP3	RHF | UHF	RHF | UHF
CCD	RHF | UHF | ROHF	RHF | UHF
LCCD	RHF | UHF | ROHF	RHF | UHF
CCSD	RHF | UHF | ROHF	RHF | UHF
CCSD(T)	RHF | UHF | ROHF	RHF | UHF
CCSD(T)$_\Lambda$	RHF | UHF | ROHF	
OMP2	RHF | UHF | ROHF	RHF | UHF
OMP2.5	RHF | UHF | ROHF	RHF | UHF
OMP3	RHF | UHF | ROHF	RHF | UHF
OLCCD	RHF | UHF | ROHF	RHF | UHF
OCCD	RHF | UHF | ROHF	RHF | UHF
OCCD(T)	RHF | UHF | ROHF	
OCCD(T)$_\Lambda$	RHF | UHF | ROHF	
FNO-MP2.5	RHF | UHF	
FNO-MP3	RHF | UHF	
FNO-CCD	RHF | UHF | ROHF	
FNO-LCCD	RHF | UHF | ROHF	
FNO-CCSD	RHF | UHF | ROHF	
FNO-CCSD(T)	RHF | UHF | ROHF	
FNO-CCSD(T)$_\Lambda$	RHF | UHF | ROHF	
EOM-CCSD	RHF | UHF	
EOM-CCD	RHF | UHF	
EOM-OCCD	RHF | UHF	
QDPT2	UHF | ROHF	
CAS-CI	UHF | ROHF	
CIS	RHF | UHF	
FCI	UHF | ROHF	
LSSMF-SCF	RHF | UHF | ROHF	
LSSMF-MP2	RHF | UHF | ROHF	
LSSMF-MP2.5	RHF | UHF	
LSSMF-MP3	RHF | UHF	
LSSMF-CCD	RHF | UHF | ROHF	
LSSMF-LCCD	RHF | UHF | ROHF	

LSSMF-CCSD	RHF | UHF | ROHF	
LSSMF-CCSD(T)	RHF | UHF | ROHF	
LSSMF-CCSD(T)$_\Lambda$	RHF | UHF | ROHF	
LSSMF-OMP2	RHF | UHF | ROHF	
LSSMF-OMP2.5	RHF | UHF | ROHF	
LSSMF-OMP3	RHF | UHF | ROHF	
LSSMF-OLCCD	RHF | UHF | ROHF	
LSSMF-OCCD	RHF | UHF | ROHF	
LSSMF-OCCD(T)	RHF | UHF | ROHF	
LSSMF-OCCD(T)$_\Lambda$	RHF | UHF | ROHF	
LSSMF-FNO-MP2.5	RHF | UHF	
LSSMF-FNO-MP3	RHF | UHF	
LSSMF-FNO-CCD	RHF | UHF | ROHF	
LSSMF-FNO-LCCD	RHF | UHF | ROHF	
LSSMF-FNO-CCSD	RHF | UHF | ROHF	
LSSMF-FNO-CCSD(T)	RHF | UHF | ROHF	
LSSMF-FNO-CCSD(T)$_\Lambda$	RHF | UHF | ROHF	

[a] For all gradient methods, ionization potentials via Extended Koopmans' Theorem (EKT) are available.

2 Citing MacroQC

2.1 Overall MacroQC Package

MacroQC, An electronic structure theory software for large-scale applications developed by U. Bozkaya with contributions from B. Ermiş, Y. Alagöz, A. Ünal, and A. K. Uyar, 2021.

2.2 Orbital-Optimized Post-Hartree-Fock Methods

2.2.1 Orbital-Optimized Second-Order Perturbation Theory (OMP2)

- Bozkaya, U. 2014. "Analytic Energy Gradients and Spin Multiplicities for Orbital-Optimized Second-Order Perturbation Theory with Density-Fitting Approximation: An Efficient Implementation". *J. Chem. Theory Comput.* 10, 4389-4399. DOI: 10.1021/ct500634s

- Bozkaya, U. 2014. "Orbital-Optimized Second-Order Perturbation Theory with Density-Fitting and Cholesky Decomposition Approximations: An Efficient Implementation". *J. Chem. Theory Comput.* 10, 2371-2378. DOI: 10.1021/ct500231c

- Bozkaya, U.; Sherrill, C. D. 2013. "Analytic Energy Gradients for the Orbital-Optimized Second-Order Møller-Plesset Perturbation Theory". *J. Chem. Phys.* 138, 184103. DOI: 10.1063/1.4803662

2.2.2 Orbital-Optimized Third-Order Perturbation Theory (OMP3) and Orbital-Optimized MP2.5 (OMP2.5)

- Bozkaya, U. 2018. "Analytic Energy Gradients for Orbital-Optimized MP3 and MP2.5 with the Density-Fitting Approximation: An Efficient Implementation". *J. Comput. Chem.* 39, 351-360. DOI: 10.1002/jcc.25122

- Bozkaya, U. 2016. "Orbital-Optimized MP3 and MP2.5 with Density-Fitting and Cholesky Decomposition Approximations". *J. Chem. Theory Comput.* 12, 1179-1188. DOI: 10.1021/acs.jctc.5b01128

- Soydaş, E.; Bozkaya, U. 2015. "Assessment of Orbital-Optimized MP2.5 for Thermochemistry and Kinetics: Dramatic Failures of Standard Perturbation Theory Approaches for Aromatic Bond Dissociation Energies and Barrier Heights of Radical Reactions". *J. Chem. Theory Comput.* 11, 1564-1573. DOI: 10.1021/ct501184w

- Bozkaya, U.; Sherrill, C. D. 2014. "Orbital-optimized MP2.5 and Its Analytic Gradients: Approaching CCSD(T) Quality for Noncovalent Interactions". *J. Chem. Phys.* 141, 204105. DOI: 10.1063/1.4902226

- Bozkaya, U. 2013. "Analytic Energy Gradients for the Orbital-Optimized Third-Order Møller-Plesset Perturbation Theory". *J. Chem. Phys.* 139, 104116. DOI: 10.1063/1.4820877

- Soydaş, E.; Bozkaya, U. 2013. "Assessment of Orbital-Optimized Third-Order Møller-Plesset Perturbation Theory and Its Spin-Component and Spin-Opposite Scaled Variants for Thermochemistry and Kinetics". *J. Chem. Theory Comput.* 9, 1452-1460. DOI: 10.1021/ct301078q

- Bozkaya, U. 2011. "Orbital-Optimized Third-Order Møller-Plesset Perturbation Theory and Its Spin-Component and Spin-Opposite Scaled Variants: Application to Symmetry Breaking Problems". *J. Chem. Phys.* 135, 224103. DOI: 10.1063/1.3665134

2.2.3 Orbital-Optimized Linearized Coupled-Cluster Doubles Method (OLCCD)

- Bozkaya, U. 2016. "Orbital-Optimized Linearized Coupled-Cluster Doubles with Density-Fitting and Cholesky Decomposition Approximations: An Efficient Implementation". Phys. Chem. Chem. Phys. 18, 11362-11373. DOI: 10.1039/C6CP00164E

- Soydaş, E.; Bozkaya, U. 2014. "Assessment of the Orbital-Optimized Coupled-Electron Pair Theory for Thermochemistry and Kinetics: Improving upon CCSD and CEPA(1)". *J. Comput. Chem.* 35 , 1073-1081. DOI: 10.1002/jcc.23592

- Bozkaya, U.; Sherrill, C. D. 2013. "Orbital-Optimized Coupled-Electron Pair Theory and its Analytic Gradients: Accurate Equilibrium Geometries, Harmonic Vibrational Frequencies, and Hydrogen Transfer Reactions". *J. Chem. Phys.* 139, 054104. DOI: 10.1063/1.4816628

2.2.4 Orbital-Optimized Coupled-Cluster Doubles Method (OCCD)

- Bozkaya, U.; Ünal, A.; Alagöz, Y. 2020. "Energy and Analytic Gradients for the Orbital-Optimized Coupled-Cluster Doubles Method with the Density-Fitting Approximation: An Efficient Implementation", *J. Chem. Phys.* 153, 244115. DOI: 10.1063/5.0035811

- Bozkaya, U.; Turney, J. M.; Yamaguchi, Y.; Schaefer, H. F.; Sherrill, C. D. 2011. "Quadratically Convergent Algorithm for Orbital Optimization in the Orbital-Optimized Coupled-Cluster Doubles Method and in Orbital-Optimized Second-Order Møller-Plesset Perturbation Theory". *J. Chem. Phys.* 135, 104103. DOI: 10.1063/1.3631129

2.2.5 Orbital-Optimized Coupled-Cluster Doubles Method with Perturbative Triples [OCCD(T) and OCCD(T)$_\Lambda$]

- Alagöz, Y.; Ünal, A.; Bozkaya, U. 2021. "Efficient Implementations of the Symmetric and Asymmetric Triple Excitation Corrections for the Orbital-Optimized Coupled-Cluster Doubles Method with the Density-Fitting Approximation", *J. Chem. Phys.* (submitted).

2.3 Perturbation Theory (PT)

- Bozkaya, U. 2018. "Analytic energy gradients for orbital-optimized MP3 and MP2.5 with the density-fitting approximation: An efficient implementation". *J. Comput. Chem.* 39, 351-360. DOI: 10.1002/jcc.25122

- Bozkaya, U. 2016. "Orbital-Optimized MP3 and MP2.5 with Density-Fitting and Cholesky Decomposition Approximations". *J. Chem. Theory Comput.* 12, 1179-1188. DOI: 10.1021/acs.jctc.5b01128

- Bozkaya, U. 2014. "Derivation of General Analytic Gradient Expressions for Density-Fitted Post-Hartree-Fock Methods: An Efficient Implementation for the Density-Fitted Second-Order Møller-Plesset Perturbation Theory". *J. Chem. Phys.* 141, 124108. DOI: 10.1063/1.4903269

- Bozkaya, U.; Sherrill, C. D. 2014. "Orbital-optimized MP2.5 and Its Analytic Gradients: Approaching CCSD(T) Quality for Noncovalent Interactions". *J. Chem. Phys.* 141, 204105. DOI: 10.1063/1.4902226

2.4 Coupled-Cluster (CC)

- Bozkaya, U.; Sherrill, C. D. 2017. "Analytic energy gradients for the coupled-cluster singles and doubles with perturbative triples method with the density-fitting approximation". *J. Chem. Phys.* 147, 044104. DOI: 10.1063/1.4994918

- Bozkaya, U. 2016. "A Noniterative Asymmetric Triple Excitation Correction for The Density-Fitted Coupled-Cluster Singles and Doubles Method: Preliminary Applications". *J. Chem. Phys.* 144, 144108. DOI: 10.1063/1.4945706

- Bozkaya, U.; Sherrill, C. D. 2016. "Analytic Energy Gradients for the Coupled-Cluster Singles and Doubles Method with the Density-Fitting Approximation". *J. Chem. Phys.* 144, 174103. DOI: 10.1063/1.4948318

- Bozkaya, U. 2014. "Derivation of General Analytic Gradient Expressions for Density-Fitted Post-Hartree-Fock Methods: An Efficient Implementation for the Density-Fitted Second-Order Møller-Plesset Perturbation Theory". *J. Chem. Phys.* 141, 124108. DOI: 10.1063/1.4903269

- Soydaş, E.; Bozkaya, U. 2014. "Assessment of the Orbital-Optimized Coupled-Electron Pair Theory for Thermochemistry and Kinetics: Improving upon CCSD and CEPA(1)". *J. Comput. Chem.* 35, 1073-1081. DOI: 10.1002/jcc.23592

- Bozkaya, U.; Sherrill, C. D. 2013. "Orbital-Optimized Coupled-Electron Pair Theory and its Analytic Gradients: Accurate Equilibrium Geometries, Harmonic Vibrational Frequencies, and Hydrogen Transfer Reactions". *J. Chem. Phys.* 139, 054104. DOI: 10.1063/1.4816628

- Bozkaya, U.; Schaefer, H. F. 2012 "Symmetric and Asymmetric Triple-Excitation Corrections for Orbital-Optimized Coupled-Cluster Doubles Method: Improving Upon CCSD(T) and CCSD(T)Λ. Application to Potential Energy Surfaces". *J. Chem. Phys.* 136, 204114. DOI: 10.1063/1.4720382

2.5 Second-Order Quasidegenerate Perturbation Theory (QDPT2)

- Servan, S. A.; Ünal, A.; Hamarat, B.; Bozkaya, U. 2020. "Assessment of the Density-Fitted Second-Order Quasidegenerate Perturbation Theory for Transition Energies: Accurate Computations of Singlet-Triplet Gaps for Charge-Transfer Compounds", *J. Phys. Chem. A* 124, 6889-6898. DOI: 10.1021/acs.jpca.0c04555

- Bozkaya, U. 2019. "An Efficient Implementation of the Second-Order Quasidegenerate Perturbation Theory with Density-Fitting and Cholesky Decomposition Approximations: Is It Possible to Use HartreeFock Orbitals for a Multiconfigurational Perturbation Theory?". *J. Chem. Theory Comput.* 15, 4415-4429.
DOI: 10.1021/acs.jctc.9b00378

2.6 Extended Koopmans' Theorem (EKT)

- Bozkaya, U.; Ünal, A. 2018. "State-of-the-Art Computations of Vertical Ionization Potentials with the Extended Koopmans' Theorem Integrated with the CCSD(T) Method". *J. Phys. Chem. A* 122, 4375-4380. DOI: 10.1021/acs.jpca.8b01851

- Bozkaya, U. 2014. "Accurate Electron Affinities from the Extended Koopmans' Theorem Based on Orbital-Optimized Methods". *J. Chem. Theory Comput.* 10, 2041-2048. DOI: 10.1021/ct500186j

- Bozkaya, U. 2013. "The Extended Koopmans' Theorem for Orbital-Optimized Methods: Accurate Computation of Ionization Potentials". *J. Chem. Phys.* 139, 154105. DOI: 10.1063/1.4825041

2.7 Molint Framework

- Bozkaya, U. 2021. "Molint 1.0: A Framework for the Computation of Molecular Integrals and Their Derivatives for Density-fitted Methods", *Int. J. Quantum Chem.* 121, e26623. DOI: 10.1002/qua.26623

2.8 Dipole Moments and One-Electron Properties

- Bozkaya, U.; Soydaş, E.; Filiz, B. 2020. "State-of-the-Art Computations of Dipole Moments Using Analytic Gradients of High-Level Density-Fitted Coupled-Cluster Methods with Focal-Point Approximations". *J. Comput. Chem.* 41, 769-779. DOI: 10.1002/jcc.26126

2.9 Anharmonic Force Field and IR Spectra

- Ermiş, B.; Ünal, A.; Soydaş, E.; Bozkaya, U. 2021. "Anharmonic Force Field from Coupled-Cluster Methods and Accurate Computation of Infrared Spectra", *Adv. Quantum Chem.* (in press).

2.10 Linear-Scaling Systematic Molecular Fragmentation

- Bozkaya, U.; Ermiş, B. 2020. Linear-Scaling Systematic Molecular Fragmentation Approach for High-Level Coupled-Cluster Methods: Coupled-Cluster Meets Macromolecules. https://doi.org/10.26434/chemrxiv.12702425.v2.

2.11 Equation-of-Motion Coupled-Cluster

- Ünal, A.; Bozkaya, U. 2022. Efficient Implementation of Equation-of-Motion Coupled-Cluster Singles and Doubles Method with the Density-Fitting Approximation: An Enhanced Algorithm for the Particle–Particle Ladder Term, *J. Chem. Theory Comput.* **18**, 1489–1500.

- Ünal, A.; Bozkaya, U. 2024. Equation-of-Motion Orbital-Optimized Coupled-Cluster Doubles Method with the Density-Fitting Approximation: An Efficient Implementation, *J. Comput. Chem.* **45**, 2969–2978.

- Ünal, A.; Bozkaya, U. 2024. Equation-of-Motion Regularized Orbital-Optimized Second-Order Perturbation Theory with the Density-Fitting Approximation, *J. Chem. Phys.* **161**, 114108.

3 Download and Installation

`MacroQC` has an easy setup with few steps. There are several alternatives for installation methods. You can choose one of them depending on your system and personal habits. These alternatives are given below. In most cases, you need nothing more than an installer. Sometimes it's not even necessary. Supported architectures are Linux x86_64 and OSX64.

3.1 Install with Conda (Linux and Mac OS X)

Conda is a cross-platform open-source package management system and environment management system. This option is one of the most popular and easy ways to install a package these days.

(1) If you don't have a Conda installer, you may get it: Conda installer.

(2) Run the following commands on a UNIX console.

```
conda install -c macroqc macroqc
```

3.2 Install with Self-Extracting Package (Linux and Mac OS X)

This method offers offline and portable installation just in one step. You need to follow the steps given below:

(1) Download MacroQC self-extracting package from this link.

(2) Run installation script at the directory where the downloaded file is located (you may need root privileges):

```
bash install-macroqc.sh
```

3.3 Manual Installation from Zip Archive (Linux and Mac OS X)

This is another alternative way to install `MacroQC`.

(1) Download `MacroQC` Zip archive from this link.

(2) Extract the Zip archive.

```
unzip macroqc.zip
```

(3) Copy files wherever you want.

4 Testing MacroQC

The test scripts and sample inputs can be found in the `MQCDATADIR/tests` directory. These tests can be run via `mqc_run_tests` script. The `MacroQC` software provides a few test options to users. `all` runs all tests, which may take a long take to complete, `quick` runs a small number of basic tests, and `dfocc` runs tests for the `dfocc` module. A summary of running test script:

- `mqc_run_tests all` — everything below; slow.
- `mqc_run_tests quick` — a small set that touches most of the code.

Any number of suite names may be given at once, for example `mqc_run_tests scf dfocc pcm`. The suites are:

suite	what it covers
`ints`	one- and two-electron integrals
`scf`	RHF, UHF, ROHF and the Kohn–Sham references
`dft`	exchange-correlation functionals
`semiempirical`	the semiempirical Hamiltonians
`oeprop`	one-electron properties and population analyses
`dfocc`	the density-fitted orbital-optimized correlation module
`lpnocc`	LPNO coupled cluster
`dlpnocc`	DLPNO coupled cluster and its gradient
`cis`	configuration interaction singles
`eomee`	equation-of-motion coupled cluster
`qdpt`	quasi-degenerate perturbation theory
`cim`	cluster-in-molecule
`mfcc`	molecular fractionation with conjugate caps
`lssmf`	linear-scaling systematic molecular fragmentation
`nbody`	many-body expansion
`opt`	geometry optimization, minima and transition states
`freq`	harmonic frequencies
`irc`	intrinsic reaction coordinate
`neb`	nudged elastic band
`fsm`	freezing string
`gsm`	growing string
`aimd`	*ab initio* molecular dynamics
`gb`	the generalized-Born model
`smx`	SM12 and the class-IV charges
`smd`	SMD
`cosmo`	COSMO
`cosmors`	COSMO-RS
`pcm`	C-PCM and IEF-PCM
`cmirs`	CMIRS
`vaspsol`	the self-consistent continuum model

Moreover, tests provide a good way to learning and practicing `MacroQC` input and output format. If a problem occurs during the tests, it can be directed to the developers. In such case please contact us from our forum (macroqc@googlegroups.com) or contact e-mail (macroqcprog@gmail.com).

5 Running `MacroQC`

5.1 Setting Environmental Variables

To run `MacroQC` executable one needs to set `MQCDATADIR` and `MQC_SCRATCH` environmental variables. During the installation procedure `MacroQC` tries to automatically detect these variables. `MQC_SCRATCH` points to the directory where `MacroQC` writes temporary binary files, while `MQCDATADIR` points to the directory where the basis set files, test inputs, and documents are located. To set these variables, one can run the following commands on a `Unix` console.

Examples for `bash`, `zsh`, and `sh`:

```
    export MQC_SCRATCH=/home/user1/custom_scrdir
    export MQCDATADIR=/home/user1/custom_datadir
```

Examples for `csh` and `tcsh`:

```
    setenv MQC_SCRATCH /home/user1/custom_scrdir
    setenv MQCDATADIR /home/user1/custom_datadir
```

5.2 Running `MacroQC` in Serial Mode

To run `MacroQC` in serial mode the user should invoke the driver of the software by simply typing `macroqc` on a Unix console. The default input file name is `input.inp` and the default output file name is `input.out`. To run an input file with a custom file name, one should type `macroqc filename.inp`. The output file will be printed as `filename.out`.

5.3 Running `MacroQC` in Parallel Mode Using OpenMP

`MacroQC` software can be run in OpenMP parallel mode. The pre-compiled binaries available at the `MacroQC` homepage support OpenMP parallel execution. To run the `MacroQC` executable with OpenMP one just need to set the environmental variables `OMP_NUM_THREADS` and `MKL_NUM_THREADS` to the number of cores that are requested. For example, in a BASH shell, one should set `export OMP_NUM_THREADS=8` and `export MKL_NUM_THREADS=8` to use 8 cores. Then the `MacroQC` program should be executed as described earlier.

5.4 Running Calculations Across Nodes Using MPI

Two kinds of calculation can be spread over several machines with MPI: fragmentation calculations, where the pieces are the fragments, and frequency calculations, where they are the displaced geometries. Both work the same way. The pieces of one job are independent of one another, so each MPI rank takes a share of them and runs each one to completion by itself. A single fragment, or a single displacement, is never split across ranks — which is what allows the rest of the program to be unaware that MPI is in use. Every rank ends up holding the same results and reaches the same total.

MPI support is not in the pre-compiled binaries; it has to be asked for at configure time, which leaves the serial build exactly as it was:

```
cmake -S . -B build-mpi -D_MPI=ON -DCMAKE_BUILD_TYPE=RELEASE
   ...
```

The recommended arrangement is **one rank per node**, with the cores of that node given to the fragment workers through `FRAG_NPROC`:

```
export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
mpirun -np <number of nodes> -ppn 1 macroqc input.inp
```

```
$set_frag
  mode run
  frag_method lssmf
  frag_nproc 12          # workers inside one rank; the cores
     of one node
$end_frag
```

One rank per node rather than one per core, because the fragmentation keeps maps of the whole molecule whose size grows as the square of the number of atoms — 0.8 GB each at ten thousand atoms. A second rank on the same node needs its own copy of them; a fragment worker, which is forked, shares its parent's. Cores are therefore better spent on `FRAG_NPROC` than on extra ranks.

Setting `FRAG_NPROC` to 1 gives the hybrid arrangement instead: one process per node, no fragment workers, and the cores used by threaded BLAS within each fragment. This suits large fragments, which thread well; the default suits many small ones, which do not.

Two practical points. The scratch directory must exist on *every* node, not only the first: either create it on each, or point `MQC_SCRATCH` at a shared filesystem, which is safe because each rank writes files named by its rank. And only rank 0 writes `<input>.out`; the other ranks write `<input>.rank`N`.out` beside it, which are kept rather than deleted, because a rank that fails is the one whose output is wanted.

Dividing the work does not change the answer. Energies and gradients are identical, to the last digit, however many ranks and workers the fragments are spread over, because each fragment is computed independently and the results are exchanged rather than accumulated.

Two things can nevertheless make two runs differ in the last few digits, and neither is the division of work. A calculation that runs without any parallel fragment machinery at all — `FRAG_NPROC` 1 on a single rank — carries each fragment's converged wavefunction forward as the initial guess for the next one, which the parallel path deliberately does not; the two can then converge to points about 10^{-12} hartree apart. And on a cluster, MKL chooses among code paths at run time, so the same arithmetic can be done in a different order from one run to the next. Setting `MKL_CBWR=COMPATIBLE` removes the second, at some cost in speed, and is worth doing when runs have to be compared digit for digit.

A frequency calculation is divided in exactly the same fashion, with `FREQ_NPROC` in place of `FRAG_NPROC`:

```
$set_freq
  mode run
  derive_level 2
  freq_nproc 12          # workers inside one rank; the cores
     of one node
$end_freq
```

Here the memory argument that favours one rank per node does not apply — a frequency job keeps nothing of the size the fragmentation maps reach — so ranks may be placed as the queue prefers. What limits the useful rank count instead is the number of displacements, $6N$

for a harmonic calculation: asking for more ranks than that leaves some of them idle. The displacements of a molecule all cost about the same, so they are handed out round-robin and no balancing is needed.

Methods with MPI parallelism of their own — CIM and DEC — run serially when they are called from inside a fragment, since the ranks are already occupied with the other fragments. Run directly, they parallelise as before.

6 Basic Input Format

`MacroQC` execution is controlled by a user-prepared input file. If the input file name is not specified on the command line, the data is read from the default input file (`input.inp`), and program results are printed to a standard output file (`input.out`). If the input file name is specified, the output file name is generated from the input file name removing any trailing suffix, and appending `.out`. If the output file already exists, `MacroQC` overwrites it.

`MacroQC` execution is controlled by a set of data and options in the input file. In general, each input record begins with a keyword, which is followed by keyword values. The command-line interface requires a `MacroQC` input file which is simply an ASCII text file. This input file can be created using your favorite editor (e.g., vim, emacs, jot, etc.) following the basic steps outlined in the next few chapters.

`MacroQC` input mechanism uses a series of section (block) keywords. The `MacroQC` program searches the input file for supported section names. When `MacroQC` finds a section name, it then reads supported keywords belong to the section. Each input section starts with `$set_section` and terminates with `$end_section`. For example, `scf` module options can be given in the `scf` block, which starts with `$set_scf` and terminates with `$end_scf`. A short description of all `MacroQC` keywords is provided in Table 2 (pp 21).

Table 2: The list of `MacroQC` input sections. The `molecule` section is only required for all jobs.

Section Start	Section End	Description
`$set_molecule`	`$end_molecule`	Contains charge, multiplicity, and the molecular coordinate input (mandatory).
`$set_layer`	`$end_layer`	Contains charge, multiplicity, and the molecular coordinate input (mandatory for `aimd` multilayer mode).
`$set_body`	`$end_body`	Contains charge, multiplicity, and the molecular coordinate input (mandatory for N-Body).
`$set_globals`	`$end_globals`	Global options.
`$set_basis`	`$end_basis`	Assign primary basis sets for individual atoms.
`$set_aux_basis_scf`	`$end_aux_basis_scf`	Assign JK-Fit type auxiliary basis sets for individual atoms.

Section Start	Section End	Description
`$set_aux_basis_corr`	`$end_aux_basis_corr`	Assign RI-type auxiliary basis sets for individual atoms.
`$set_sad_aux_basis`	`$end_sad_aux_basis`	Assign the auxiliary basis set for the SAD guess.
`$set_opt`	`$end_opt`	Options for the `opt` module.
`$set_freq`	`$end_freq`	Options for the `freq` module.
`$set_scf`	`$end_scf`	Options for the `scf` module.
`$set_dfocc`	`$end_dfocc`	Options for the `dfocc` module.
`$set_qdpt`	`$end_qdpt`	Options for the `qdpt` module.
`$set_eomee`	`$end_eomee`	Options for the `eomee` module.
`$set_frag`	`$end_frag`	Options for the `fragment` module.
`$set_lssmf`	`$end_lssmf`	Options for the LSSMF.
`$set_nbody`	`$end_nbody`	Options for the N-Body.
`$set_mfcc`	`$end_mfcc`	Options for the MFCC.
`$set_aimd`	`$end_aimd`	Options for the `aimd` module.
`$set_external`	`$end_external`	Options for the external software used for `fragment` or `freq` modules.
`$set_basis2`	`$end_basis2`	Assign the secondary basis set for individual atoms.
`$set_ints`	`$end_ints`	Options for the integrals.
`$set_trans`	`$end_trans`	Options for the integral transformation.
`$set_cis`	`$end_cis`	Options for the CIS module.
`$set_lpnocc`	`$end_lpnocc`	Options for the LPNO and DLPNO modules.
`$set_pno`	`$end_pno`	Options for the pair natural orbitals.
`$set_cim`	`$end_cim`	Options for the cluster-in-molecule method.
`$set_dec`	`$end_dec`	Options for the divide-expand-consolidate method.
`$set_scan`	`$end_scan`	Options for a coordinate scan.
`$set_irc`	`$end_irc`	Options for the intrinsic reaction coordinate.
`$set_neb`	`$end_neb`	Options for the nudged elastic band.
`$set_fsm`	`$end_fsm`	Options for the freezing string method.
`$set_gsm`	`$end_gsm`	Options for the growing string method.
`$set_numquad`	`$end_numquad`	Options for the numerical quadrature.
`$set_ml`	`$end_ml`	Options for the machine-learning module.
`$set_paramfit`	`$end_paramfit`	Options for parameter fitting.
`$set_plugin`	`$end_plugin`	Options for plugins.
`$set_test`	`$end_test`	Options for the tests.
solvation		
`$set_gepol`	`$end_gepol`	Options for the cavity and the surface tessellation.
`$set_gb`	`$end_gb`	Options for the generalized-Born model.
`$set_smx`	`$end_smx`	Options for SM12 and the class-IV charges.
`$set_smd`	`$end_smd`	Options for the SMD model.
`$set_cosmo`	`$end_cosmo`	Options for COSMO.
`$set_cosmors`	`$end_cosmors`	Options for COSMO-RS.
`$set_pcm`	`$end_pcm`	Options for C-PCM and IEF-PCM.
`$set_cmirs`	`$end_cmirs`	Options for CMIRS.
`$set_vaspsol`	`$end_vaspsol`	Options for the self-consistent continuum model.

* Note that users can enter keyword sections in any order.

* Note that after `$set_molecule` user may add a name such as `$set_molecule water`. This feature is valid only for the `molecule` section.

* Note that the molecular geometry input may be given either in Cartesian coordinates or as a Z-matrix (see Section 6.3), and atom symbols should be used in the geometry section.

6.1 Input Examples

*Single-point energy computation at the `rhf/cc-pvdz` level:

```
# This line is a comment line
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    method scf
    reference rhf
    jobtype energy
    memory 2000
$end_globals
```

6.2 Ghost Atoms

Writing the `@` symbol before the atom symbol of any atom in the `molecule` section will make that atom a ghost atom. Ghost atoms possess basis functions but they do not have electrons or nuclear charges. The ghost atoms can only be used for energy computations. *Single-point energy computation at the `rhf/cc-pvdz` level:

```
$set_molecule H2O-NH3
 0 1
 O   0.00000   0.00000  -0.06577
 H   0.00000  -0.75906   0.52195
 H   0.00000   0.75906   0.52195
 @N  0.00000   0.01436   1.46454
 @H  0.00000  -0.98104   1.65344
 @H -0.81348   0.39876   1.92934
 @H  0.81348   0.39876   1.92934
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  method scf
  reference rhf
  jobtype energy
```

```
$end_globals
```

6.3 Z-Matrix Input

The molecular geometry may be given as a Z-matrix instead of Cartesian coordinates. The format is recognized automatically, so no keyword is needed; the two formats must not be mixed within one `molecule` section. Each line places one center relative to the ones above it,

```
Symbol  i  R  j  A  k  D
```

where `i`, `j` and `k` are the numbers (counting from 1, in the order written) of three earlier and mutually distinct centers, `R` is the distance to center `i`, `A` is the `i`-centered bond angle to `j`, and `D` is the `k`-`j`-`i`-`Symbol` dihedral angle. The first three lines carry fewer fields: the first center has only a symbol, the second only `i R`, and the third only `i R j A`. Bond lengths follow the `geom_units` keyword (Ångström by default), whereas angles are always in degrees. The dihedral follows the usual convention: it is zero when the fourth center eclipses center `k`, and positive for a clockwise rotation of the `i`-`Symbol` bond with respect to the `j`-`k` bond when viewed along `j`→`i`.

Values may also be given as named variables, defined anywhere in the section as `NAME = VALUE`. A variable may be used with a leading minus sign to negate it, which is convenient for pairs of torsions.

```
$set_molecule HOOH
  0 1
  O
  O 1 ROO
  H 1 ROH 2 AOOH
  H 2 ROH 1 AOOH 3 -TAU

  ROO  = 1.45
  ROH  = 0.97
  AOOH = 100.0
  TAU  = 113.7
$end_molecule
```

A center whose symbol is `X` (or `XX`) is a dummy center. Dummy centers carry no nuclear charge, no electrons, and no basis functions; they exist only to define the geometry and are removed once the Cartesian coordinates have been built. They are the standard way to handle linear fragments, whose 180° angles leave a torsion undefined.

```
$set_molecule C2H2
  0 1
  C
  C 1 1.203
  X 1 1.0   2 90.0
```

```
  H 1 1.061 3 90.0  2 180.0
  H 2 1.061 1 180.0 3 0.0
$end_molecule
```

Ghost atoms are written as in Cartesian input, by placing `@` before the atom symbol. A ghost center may appear anywhere in the Z-matrix and may be used as a reference by later centers. Note that, as with Cartesian input, the ghost atoms are moved to the end of the atom list, so the atom numbering printed in the output need not follow the order of the Z-matrix lines.

```
$set_molecule H2O-DIMER-CP
  0 1
  O
  @O 1 2.90
  H  1 0.96 2 100.0
  H  1 0.96 2 100.0 3 104.5
  @H 2 0.96 1 100.0 3 60.0
$end_molecule
```

The Cartesian coordinates built from the Z-matrix are printed in the output file, and everything downstream (optimization, frequencies, AIMD, fragmentation) then proceeds exactly as for Cartesian input.

* Note that external charges (`EC`) are not available in Z-matrix input; use Cartesian coordinates for those.

6.4 External Charges

The format for external charges is the `EC` symbol, the Cartesian coordinates, followed by the charge. External charges are also shifted to the center of mass with the molecule.
*Single-point energy computation at the `rhf/cc-pvdz` level:

```
$set_molecule
1 1
 O    1.46710223    -0.96026401    -0.84235688
 H    2.38494062    -0.65229087    -0.74535997
 H    1.50141665    -1.75285797    -1.39080538
 O   -1.45264971     0.60446441    -1.14352866
 H   -1.47721698     1.15725057    -1.93346483
 H   -2.37258328     0.34732430    -0.95837892
 O    1.14694134     1.35761780     0.78995323
 H    0.96540138     2.12759124     1.34163941
 H    2.11027535     1.33621950     0.65502051
 O   -1.16127440    -1.00259910     1.19567449
 H   -0.98941492    -1.53327044     1.98245878
 H   -2.12261877    -1.03132143     1.04846012
 O    3.74848604     0.57050669    -0.07526597
```

```
 H    4.25830799     1.09057781    -0.71369648
 H    4.41479872     0.22966949     0.53942387
 O   -3.74861116    -0.56969516     0.07555459
 H   -4.25143300    -1.27985843    -0.34967876
 H   -4.42204079    -0.03797115     0.52471820
 EC   0.00003690    -0.00022249    -0.00011859 1.0
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  method scf
  reference rhf
  jobtype energy
$end_globals
```

6.5 Multiple Jobs

Multiple jobs can be combined into a single input file. The `@@@` string line is used to separate jobs. All results are appended to a single output file. If you want to use the same options for different geometries, you should give the options in the first job. It is also possible to run completely independent jobs in the same input.

*Single-point energy computation at the `mp2/cc-pvdz` level for multiple molecular geometries using a single option block:

```
$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    method scf
    reference rhf
    jobtype energy
$end_globals

$set_molecule H2O
    0 1
    O 0.00000  0.00000 -0.06577
    H 0.00000 -0.75906  0.52195
    H 0.00000  0.75906  0.52195
$end_molecule

@@@

$set_molecule CH4
    0 1
```

```
    C 1.06474  0.20611  0.30975
    H 0.84383  0.16523  1.38297
    H 0.98386  1.24826 -0.00369
    H 0.38021 -0.42906 -0.24139
    H 2.07622 -0.15282  0.18373
$end_molecule

@@@
```

*Completely independent jobs in the same input.

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    method scf
    reference rhf
    jobtype energy
$end_globals

@@@

$set_molecule CH4
    0 1
    C   1.06474   0.20611   0.30975
    H   0.84383   0.16523   1.38297
    H   0.98386   1.24826  -0.00369
    H   0.38021  -0.42906  -0.24139
    H   2.07622  -0.15282   0.18373
$end_molecule

$set_globals
    basis cc-pvtz
    aux_basis_scf cc-pvtz-jkfit
    aux_basis_corr cc-pvtz-ri
    method mp2
    reference rhf
    jobtype opt
$end_globals
```

```
@@@
```

6.6 Global Options

- **ERI_DISK**
  Writes electron-repulsion integral data to disk when supported by the selected method.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **FREEZE_CORE**
  Do apply frozen core approximation for post-Hartree–Fock computations? See frozen core table for individual atoms.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **MOL_FRAG**
  Do use molecular fragmentation techniques? See `fragment` module options if this option is set to TRUE.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **NOCOM**
  Do not shift molecular geometry to the center of mass?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **NOREORIENT**
  Do not reorient molecular geometry?
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **PUREAM**
  Do use pure angular momentum functions in molecular integral evaluations? The `molint` library is optimized for spherical functions. The usage of Cartesian functions is experimental. Hence, we recommend using the default option.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **READ_PDB**
  Reads the molecular structure from a PDB-style input. This option is also activated

automatically when a PDB file is detected by the parser.
TYPE: Boolean
DEFAULT: FALSE
OPTIONS: TRUE, FALSE

- **TEST_MODE**
  Activates test mode.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **FINDIF_POINTS**
  The number of points used in finite difference formulas.
  TYPE: Integer
  DEFAULT: 3
  OPTIONS: 3, 5

- **MEMORY**
  Amount of total memory supplied to the software in MB.
  TYPE: Integer
  DEFAULT: 2000
  OPTIONS: n User-defined value.

- **NUM_THREADS**
  Number of computational threads to be used where thread parallelism is available.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **EEQ_CHARGES**
  Print the charge-equilibration (EEQ2019) atomic charges and coordination numbers of the molecule. The model needs no wavefunction — it is a function of the elements and the geometry alone — so the charges are reported before anything else runs, for any `JOBTYPE`. At `PRINT_LEVEL` 2 and above the analytic gradient of $\frac{1}{2}\sum_i q_i^2$ is printed as well, as a self-test of the charge derivative.
  TYPE: Bool
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **EC_EC_REPULSION**
  Whether the mutual Coulomb repulsion of the external point charges (`EC` lines in the molecule block) belongs in the energy. For a molecule sitting in a field of charges that are not its own atoms it does, and the default keeps it. For a fragment embedded in the charges of its own surroundings it does not: that is the environment interacting with itself, it is not part of the fragment's energy, and because every fragment sees a different environment it does not cancel in the fragment sum either. LSSMF therefore turns this off for itself.

TYPE: Bool
DEFAULT: TRUE
OPTIONS: TRUE, FALSE

- **NUM_FROZEN_DOCC**
  The Number of frozen doubly occupied orbitals to be used in post-Hartree–Fock computations. This option overwrites the FREEZE_CORE option.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **NUM_FROZEN_VIR**
  Number of frozen virtuals to be used in post-Hartree–Fock computations.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **PRINT_LEVEL**
  This option controls the amount of information to print to the output file.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **OUTFILE_MODE**
  Controls how the main output file is opened.
  TYPE: String
  DEFAULT: NEW
  OPTIONS: NEW, APPEND

  - NEW: Creates a new output file.
  - APPEND: Appends new results to an existing output file.

- **STEP_BOHR**
  Specifies the step-size for numerical differentiation (in atomic units). The same value should be used for the `SOW` and `REAP` options of `freq` module: the Hessian is formed as $[g(+h) - g(-h)]/2h$, so reaping gradients that were sown with a different h rescales every frequency by $\sqrt{h_{\mathrm{sow}}/h_{\mathrm{reap}}}$. The default matches the one used by Psi4; enlarging it lets anharmonicity leak into the numerical second derivative, which shifts stretching frequencies by several cm^{-1}.
  TYPE: Double
  DEFAULT: 0.005
  OPTIONS: n User-defined value.

- **AUX_BASIS_CORR**
  Auxiliary basis set to be used in electron correlation computations.
  TYPE: String

DEFAULT: CC-PVDZ-RI
OPTIONS: User defined basis set.

- **AUX_BASIS_SCF**
  Auxiliary basis set to be used in SCF computations.
  TYPE: String
  DEFAULT: CC-PVDZ-JKFIT
  OPTIONS: User defined basis set.

- **BASIS**
  Primary basis set to be used in computations.
  TYPE: String
  DEFAULT: CC-PVDZ
  OPTIONS: User defined basis set.

- **BASIS2**
  Secondary basis set used by modules that require an additional auxiliary or minimal basis. The current default is `minao`.
  TYPE: String
  DEFAULT: MINAO
  OPTIONS: User defined basis set.

- **DERIVE_TYPE**
  Type of derivatives to be used.
  TYPE: String
  DEFAULT: ANALYTIC
  OPTIONS: ANALYTIC, NUMERIC

- **FINDIF_METHOD**
  The method used in FINDIF formulas.
  TYPE: String
  DEFAULT: GRAD
  OPTIONS: GRAD, ENERGY

- **GEOM_UNITS**
  The unit of molecular geometry.
  TYPE: String
  DEFAULT: ANG
  OPTIONS: ANG, BOHR

- **JOBTYPE**
  Type of job that requested from the software.
  TYPE: String
  DEFAULT: ENERGY
  OPTIONS: ENERGY, GRAD, OPT, FREQ, IRC, SCAN, FSM, NEB, INPUT, AIMD, PARAMFIT, ML

  – ENERGY: Single-point energy job.

- GRAD: Analytic gradients job.
- OPT: Geometry optimization job.
- FREQ: Vibrational frequency job.
- IRC: Intrinsic reaction coordinate job.
- SCAN: Potential energy surface scan along one internal coordinate; PES_SCAN is a synonym.
- FSM: Freezing string method — a transition-state guess, and the saddle search from it, built from the reactant and the product.
- NEB: Nudged elastic band — the whole minimum energy path between the reactant and the product, and, with the climbing image, the saddle point on it.
- INPUT: Checks the syntax, charge, and multiplicity without any computations.
- AIMD: ab-initio molecular dynamic simulations job.
- PARAMFIT: Parameter-fitting workflow.
- ML: Machine-learning workflow.

- **REFERENCE**
  The reference wave function for SCF computations.
  TYPE: String
  DEFAULT: RHF
  OPTIONS: RHF, UHF, ROHF

  - RHF: Restricted closed-shell Hartree–Fock.
  - UHF: Unrestricted Hartree–Fock.
  - ROHF: Restricted open-shell Hartree–Fock.

- **METHOD**
  A Quantum chemical method will be used.
  TYPE: String
  DEFAULT: SCF
  OPTIONS: SCF, MP2, OMP2, MP2.5, OMP2.5, MP3, OMP3, CIS, CCD, OCCD, OCCD(T), OCCD(AT), LCCD, OLCCD, CCSD, CCSD(T), CCSD(AT), QDPT2, CAS-CI, FCI, LPNO-CCSD, LPNO-CCSD(T), EOM-CCSD, EOM-CCD, EOM-OCCD, EOM-MP2, EOM-OMP2, EOM-MP3, EOM-OMP3, EOM-MP2.5, EOM-OMP2.5, EOM-LCCD, EOM-OLCCD, CIM-MP2, CIM-MP2.5, CIM-MP3, CIM-CCSD, CIM-CCSD(T), DEC-MP2, DEC-CCSD, CIM-EOM-MP2, CIM-EOM-CCSD, CIM-FNO-EOM-MP2, CIM-FNO-EOM-CCSD, DEC-EOM-MP2, DEC-EOM-CCSD

  - SCF: Self-consistent field method.
  - MP2: Second-order Møller–Plesset perturbation theory.
  - OMP2: Orbital-optimized second-order Møller–Plesset perturbation theory.
  - MP2.5: The MP2.5 model.

- OMP2.5: Orbital-optimized MP2.5 model.
- MP3: Third-order Møller–Plesset perturbation theory.
- OMP3: Orbital-optimized third-order Møller–Plesset perturbation theory.
- CIS: Configuration interaction singles.
- CCD: Coupled-cluster doubles.
- OCCD: Orbital-optimized coupled-cluster doubles.
- OCCD(T): Orbital-optimized coupled-cluster doubles with perturbative triples.
- OCCD(AT): Orbital-optimized coupled-cluster doubles with asymmetric perturbative triples.
- LCCD: Linearized coupled-cluster doubles.
- OLCCD: Orbital-optimized linearized coupled-cluster doubles.
- CCSD: Coupled-cluster singles and doubles.
- CCSD(T): Coupled-cluster singles and doubles with perturbative triples.
- CCSD(AT): Coupled-cluster singles and doubles with asymmetric perturbative triples.
- CAS-CI: Complete active space configuration interaction.
- FCI: Full configuration interaction.
- LPNO-CCSD: Local pair-natural orbital CCSD.
- LPNO-CCSD(T): Local pair-natural orbital CCSD with perturbative triples.
- EOM-CCSD: Equation-of-motion coupled-cluster singles and doubles.
- EOM-CCD: Equation-of-motion coupled-cluster doubles.
- EOM-OCCD: Equation-of-motion orbital-optimized coupled-cluster doubles.
- Additional EOM, CIM, CIM-FNO, and DEC variants are available for the corresponding methods listed above. See the relevant module sections for details.

- **ELECTRIC_FIELD**
  Applies a uniform external electric field during the SCF computation.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, X, Y, Z, XY, XZ, YZ, XYZ

- **ELECTRIC_STRENGTH**
  Cartesian field strengths associated with `ELECTRIC_FIELD`. Three values are supplied for the x, y, and z components, respectively.
  TYPE: Double Array
  DEFAULT: [0.001;0.001;0.001]
  OPTIONS: User-defined three-component array.

- **ALIGN_TWO_ATOMS**
  Aligns a selected atom pair along a Cartesian axis before the computation. The atom indices are provided through the `TWO_ATOMS` option.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, X, Y, Z

- **TWO_ATOMS**
  Defines the two atom indices used by the `ALIGN_TWO_ATOMS` option.
  TYPE: Integer Array
  DEFAULT: [0]
  OPTIONS: User-defined two-element array.

6.7 Default Values for the Frozen Core Approximation

In electron correlation computations only the valence electrons are generally considered. The remaining electrons are kept frozen, which defines a frozen core. Table 3 presents default values of frozen core orbitals for post-HF computations. These values can be overwritten by user-defined values using the `NUM_FROZEN_DOCC` option.

Table 3: Default values for number of frozen core orbitals.

0 H																	0 He
0 Li	0 Be											1 B	1 C	1 N	1 O	1 F	1 Ne
1 Na	1 Mg											5 Al	5 Si	5 P	5 S	5 Cl	5 Ar
5 K	5 Ca	5 Sc	5 Ti	5 V	5 Cr	5 Mn	5 Fe	5 Co	5 Ni	5 Cu	5 Zn	9 Ga	9 Ge	9 As	9 Se	9 Br	9 Kr
9 Rb	9 Sr	14 Y	14 Zr	14 Nb	14 Mo	14 Tc	14 Ru	14 Rh	14 Pd	14 Ag	14 Cd	18 In	18 Sn	18 Sb	18 Te	18 I	18 Xe
18 Cs	18 Ba	23 Lu	23 Hf	23 Ta	23 W	23 Re	23 Os	23 Ir	23 Pt	23 Au	23 Hg	34 Tl	34 Pb	34 Bi	34 Po	34 At	34 Rn
34 Fr	34 Ra	34 Lr	50 Rf	50 Db	50 Sg	50 Bh	50 Hs	50 Mt	50 Ds	50 Rg	50 Cn	50 Nh	50 Fl	50 Mc	50 Lv	50 Ts	50 Og

Lanthanides	18 La	18 Ce	18 Pr	18 Nd	18 Pm	18 Sm	18 Eu	18 Gd	18 Tb	18 Dy	18 Ho	18 Er	18 Tm	18 Yb
Actinides	34 Ac	34 Th	34 Pa	34 U	34 Np	34 Pu	34 Am	34 Cm	34 Bk	34 Cf	34 Es	34 Fm	34 Md	34 No

Please note that Dunning's cc-pVXZ (X=2-6) basis sets [21, 22] are designed for frozen core computations. Hence, in the case of all-electron computations, it is better to use the cc-pCVXZ or cc-pwCVXZ (X=2-6) sets instead.

7 Basis Sets

7.1 Overview

A basis set is a set of functions used to generate molecular orbitals by a linear combination of basis functions. `MacroQC` uses atom-centered orbitals as basis sets, in short, atomic orbitals (AO). `MacroQC` employs contracted Gaussian functions as basis functions. By default, `MacroQC` generates spherical functions, more specifically real-valued solid-harmonics. The usage of Cartesian functions are also supported for experimental purposes.

The general form of unnormalized primitive Cartesian Gaussian type orbitals (GTOs) centered at R_A can be written as

$$G_{ijk}(a, \boldsymbol{r_A}) \;=\; \big(x - A_x\big)^i \big(y - A_y\big)^j \big(z - A_z\big)^k e^{-a\boldsymbol{r_A^2}}, \tag{1}$$

where i, j, k are the x, y, z components of the angular momentum, respectively, and $\boldsymbol{r}$ is the position vector.

A real-valued spherical GTO with quantum numbers l and m, with exponent a, centered at $\mathbf{A}$ is given by,

$$G_{lm}(\mathbf{r}, a, \mathbf{A}) \;=\; S_{lm}(x_A, y_A, z_A) e^{-a\boldsymbol{r_A^2}}, \tag{2}$$

where $S_{lm}(\mathbf{r_A})$ are the real solid-harmonics [20, 23]. The spherical GTOs can be obtained from the Cartesian GTOs by a simple transformation [23, 24].

Contracted Gaussian type orbitals (CGTOs) can be written as

$$G_{\mu lm}(\mathbf{r}, a, \mathbf{A}) \;=\; \sum_p d_{\mu p} G_{lm}(\mathbf{r}, a_p, \mathbf{A}), \tag{3}$$

where $d_{\mu p}$ is the contraction coefficient.

7.2 Built-In Basis Sets

The built-in primary basis sets and their default `JK-Fit` and `RI-Fit` auxiliary basis sets in the `MacroQC` software are presented in Table 5. In the input file, the primary basis set is defined with the `basis` option, while `JK-Fit` [6] and `RI-Fit` [25] basis sets are defined with `aux_basis_scf` and `aux_basis_corr` options, respectively. All these options should be written in the `globals` block. Hence, the user may change default auxiliary basis sets with these options. Further, in the input file, the names given in the **Basis** column of Table 5 should be written. For example, for the cc-pV(D+d)Z basis set, one needs to set `basis cc-pv_dpd_z`.

Angular momentum convention that used in the `basis` library of the `MacroQC` program is reported in Table 4.

Table 4: Angular momentum convention used in `MacroQC`.

L	0	1	2	3	4	5	6	7	8	9	10
Orbital	s	p	d	f	g	h	i	k	l	m	n

Table 5: Built-In Basis Sets in `MacroQC`.

Basis Family	**Basis**	**JK Auxiliary Basis**	**MP2 (RI) Auxiliary Basis**
cc-pVDZ	cc-pvdz	cc-pvdz-jkfit	cc-pvdz-ri
cc-pV(D+d)Z	cc-pv_dpd_z	cc-pvdz-jkfit	cc-pvdz-ri
cc-pCVDZ	cc-pcvdz	-	-
cc-pCV(D+d)Z	cc-pcv_dpd_z	-	-
cc-pwCVDZ	cc-pwcvdz	cc-pvdz-jkfit	cc-pwcvdz-ri
cc-pwCV(D+d)Z	cc-pwcv_dpd_z	cc-pvdz-jkfit	cc-pwcvdz-ri
aug-cc-pVDZ	aug-cc-pvdz	aug-cc-pvdz-jkfit	aug-cc-pvdz-ri
aug-cc-pV(D+d)Z	aug-cc-pv_dpd_z	aug-cc-pvdz-jkfit	aug-cc-pvdz-ri
aug-cc-pCVDZ	aug-cc-pcvdz	-	-
aug-cc-pCV(D+d)Z	aug-cc-pcv_dpd_z	-	-
aug-cc-pwCVDZ	aug-cc-pwcvdz	aug-cc-pvdz-jkfit	aug-cc-pwcvdz-ri
aug-cc-pwCV(D+d)Z	aug-cc-pwcv_dpd_z	aug-cc-pvdz-jkfit	aug-cc-pwcvdz-ri
heavy-aug-cc-pVDZ	heavy-aug-cc-pvdz	heavy-aug-cc-pvdz-jkfit	heavy-aug-cc-pvdz-ri
heavy-aug-cc-pV(D+d)Z	heavy-aug-cc-pv_dpd_z	heavy-aug-cc-pvdz-jkfit	heavy-aug-cc-pvdz-ri
heavy-aug-cc-pCVDZ	heavy-aug-cc-pcvdz	-	-
heavy-aug-cc-pCV(D+d)Z	heavy-aug-cc-pcv_dpd_z	-	-
heavy-aug-cc-pwCVDZ	heavy-aug-cc-pwcvdz	heavy-aug-cc-pvdz-jkfit	heavy-aug-cc-pwcvdz-ri
heavy-aug-cc-pwCV(D+d)Z	heavy-aug-cc-pwcv_dpd_z	heavy-aug-cc-pvdz-jkfit	heavy-aug-cc-pwcvdz-ri
jun-cc-pVDZ	jun-cc-pvdz	jun-cc-pvdz-jkfit	jun-cc-pvdz-ri
jun-cc-pV(D+d)Z	jun-cc-pv_dpd_z	jun-cc-pvdz-jkfit	jun-cc-pvdz-ri
jun-cc-pCVDZ	jun-cc-pcvdz	-	-
jun-cc-pCV(D+d)Z	jun-cc-pcv_dpd_z	-	-
jun-cc-pwCVDZ	jun-cc-pwcvdz	jun-cc-pvdz-jkfit	jun-cc-pwcvdz-ri
jun-cc-pwCV(D+d)Z	jun-cc-pwcv_dpd_z	jun-cc-pvdz-jkfit	jun-cc-pwcvdz-ri
d-aug-cc-pVDZ	d-aug-cc-pvdz	-	-
d-aug-cc-pCVDZ	d-aug-cc-pcvdz	-	-
d-aug-cc-pwCVDZ	d-aug-cc-pwcvdz	-	-
cc-pVDZ-DK	cc-pvdz-dk	-	-
cc-pCVDZ-DK	cc-pcvdz-dk	-	-
aug-cc-pVDZ-DK	aug-cc-pvdz-dk	-	-
aug-cc-pCVDZ-DK	aug-cc-pcvdz-dk	-	-
heavy-aug-cc-pVDZ-DK	heavy-aug-cc-pvdz-dk	-	-
heavy-aug-cc-pCVDZ-DK	heavy-aug-cc-pcvdz-dk	-	-
cc-pVTZ	cc-pvtz	cc-pvtz-jkfit	cc-pvtz-ri
cc-pV(T+d)Z	cc-pv_tpd_z	cc-pvtz-jkfit	cc-pvtz-ri

cc-pCVTZ	cc-pcvtz	-	-
cc-pCV(T+d)Z	cc-pcv_tpd_z	-	-
cc-pwCVTZ	cc-pwcvtz	cc-pvtz-jkfit	cc-pwcvtz-ri
cc-pwCV(T+d)Z	cc-pwcv_tpd_z	cc-pvtz-jkfit	cc-pwcvtz-ri
aug-cc-pVTZ	aug-cc-pvtz	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
aug-cc-pV(T+d)Z	aug-cc-pv_tpd_z	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
aug-cc-pCVTZ	aug-cc-pcvtz	-	-
aug-cc-pCV(T+d)Z	aug-cc-pcv_tpd_z	-	-
aug-cc-pwCVTZ	aug-cc-pwcvtz	aug-cc-pvtz-jkfit	aug-cc-pwcvtz-ri
aug-cc-pwCV(T+d)Z	aug-cc-pwcv_tpd_z	aug-cc-pvtz-jkfit	aug-cc-pwcvtz-ri
heavy-aug-cc-pVTZ	heavy-aug-cc-pvtz	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
heavy-aug-cc-pV(T+d)Z	heavy-aug-cc-pv_tpd_z	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
heavy-aug-cc-pCVTZ	heavy-aug-cc-pcvtz	-	-
heavy-aug-cc-pCV(T+d)Z	heavy-aug-cc-pcv_tpd_z	-	-
heavy-aug-cc-pwCVTZ	heavy-aug-cc-pwcvtz	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pwcvtz-ri
heavy-aug-cc-pwCV(T+d)Z	heavy-aug-cc-pwcv_tpd_z	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pwcvtz-ri
jun-cc-pVTZ	jun-cc-pvtz	jun-cc-pvtz-jkfit	jun-cc-pvtz-ri
jun-cc-pV(T+d)Z	jun-cc-pv_tpd_z	jun-cc-pvtz-jkfit	jun-cc-pvtz-ri
jun-cc-pCVTZ	jun-cc-pcvtz	-	-
jun-cc-pCV(T+d)Z	jun-cc-pcv_tpd_z	-	-
jun-cc-pwCVTZ	jun-cc-pwcvtz	jun-cc-pvtz-jkfit	jun-cc-pwcvtz-ri
jun-cc-pwCV(T+d)Z	jun-cc-pwcv_tpd_z	jun-cc-pvtz-jkfit	jun-cc-pwcvtz-ri
may-cc-pVTZ	may-cc-pvtz	may-cc-pvtz-jkfit	may-cc-pvtz-ri
may-cc-pV(T+d)Z	may-cc-pv_tpd_z	may-cc-pvtz-jkfit	may-cc-pvtz-ri
may-cc-pCVTZ	may-cc-pcvtz	-	-
may-cc-pCV(T+d)Z	may-cc-pcv_tpd_z	-	-
may-cc-pwCVTZ	may-cc-pwcvtz	may-cc-pvtz-jkfit	may-cc-pwcvtz-ri
may-cc-pwCV(T+d)Z	may-cc-pwcv_tpd_z	may-cc-pvtz-jkfit	may-cc-pwcvtz-ri
d-aug-cc-pVTZ	d-aug-cc-pvtz	-	-
d-aug-cc-pCVTZ	d-aug-cc-pcvtz	-	-
d-aug-cc-pwCVTZ	d-aug-cc-pwcvtz	-	-
cc-pVTZ-DK	cc-pvtz-dk	-	-
cc-pCVTZ-DK	cc-pcvtz-dk	-	-
cc-pwCVTZ-DK	cc-pwcvtz-dk	-	-
aug-cc-pVTZ-DK	aug-cc-pvtz-dk	-	-
aug-cc-pCVTZ-DK	aug-cc-pcvtz-dk	-	-
aug-cc-pwCVTZ-DK	aug-cc-pwcvtz-dk	-	-
heavy-aug-cc-pVTZ-DK	heavy-aug-cc-pvtz-dk	-	-
heavy-aug-cc-pCVTZ-DK	heavy-aug-cc-pcvtz-dk	-	-
heavy-aug-cc-pwCVTZ-DK	heavy-aug-cc-pwcvtz-dk	-	-
cc-pVQZ	cc-pvqz	cc-pvqz-jkfit	cc-pvqz-ri
cc-pV(Q+d)Z	cc-pv_qpd_z	cc-pvqz-jkfit	cc-pvqz-ri
cc-pCVQZ	cc-pcvqz	-	-
cc-pCV(Q+d)Z	cc-pcv_qpd_z	-	-
cc-pwCVQZ	cc-pwcvqz	cc-pvqz-jkfit	cc-pwcvqz-ri
cc-pwCV(Q+d)Z	cc-pwcv_qpd_z	cc-pvqz-jkfit	cc-pwcvqz-ri
aug-cc-pVQZ	aug-cc-pvqz	aug-cc-pvqz-jkfit	aug-cc-pvqz-ri

aug-cc-pV(Q+d)Z	aug-cc-pv_qpd_z	aug-cc-pvqz-jkfit	aug-cc-pvqz-ri
aug-cc-pCVQZ	aug-cc-pcvqz	-	-
aug-cc-pCV(Q+d)Z	aug-cc-pcv_qpd_z	-	-
aug-cc-pwCVQZ	aug-cc-pwcvqz	aug-cc-pvqz-jkfit	aug-cc-pwcvqz-ri
aug-cc-pwCV(Q+d)Z	aug-cc-pwcv_qpd_z	aug-cc-pvqz-jkfit	aug-cc-pwcvqz-ri
heavy-aug-cc-pVQZ	heavy-aug-cc-pvqz	heavy-aug-cc-pvqz-jkfit	heavy-aug-cc-pvqz-ri
heavy-aug-cc-pV(Q+d)Z	heavy-aug-cc-pv_qpd_z	heavy-aug-cc-pvqz-jkfit	heavy-aug-cc-pvqz-ri
heavy-aug-cc-pCVQZ	heavy-aug-cc-pcvqz	-	-
heavy-aug-cc-pCV(Q+d)Z	heavy-aug-cc-pcv_qpd_z	-	-
heavy-aug-cc-pwCVQZ	heavy-aug-cc-pwcvqz	heavy-aug-cc-pvqz-jkfit	heavy-aug-cc-pwcvqz-ri
heavy-aug-cc-pwCV(Q+d)Z	heavy-aug-cc-pwcv_qpd_z	heavy-aug-cc-pvqz-jkfit	heavy-aug-cc-pwcvqz-ri
jun-cc-pVQZ	jun-cc-pvqz	jun-cc-pvqz-jkfit	jun-cc-pvqz-ri
jun-cc-pV(Q+d)Z	jun-cc-pv_qpd_z	jun-cc-pvqz-jkfit	jun-cc-pvqz-ri
jun-cc-pCVQZ	jun-cc-pcvqz	-	-
jun-cc-pCV(Q+d)Z	jun-cc-pcv_qpd_z	-	-
jun-cc-pwCVQZ	jun-cc-pwcvqz	jun-cc-pvqz-jkfit	jun-cc-pwcvqz-ri
jun-cc-pwCV(Q+d)Z	jun-cc-pwcv_qpd_z	jun-cc-pvqz-jkfit	jun-cc-pwcvqz-ri
may-cc-pVQZ	may-cc-pvqz	may-cc-pvqz-jkfit	may-cc-pvqz-ri
may-cc-pV(Q+d)Z	may-cc-pv_qpd_z	may-cc-pvqz-jkfit	may-cc-pvqz-ri
may-cc-pCVQZ	may-cc-pcvqz	-	-
may-cc-pCV(Q+d)Z	may-cc-pcv_qpd_z	-	-
may-cc-pwCVQZ	may-cc-pwcvqz	may-cc-pvqz-jkfit	may-cc-pwcvqz-ri
may-cc-pwCV(Q+d)Z	may-cc-pwcv_qpd_z	may-cc-pvqz-jkfit	may-cc-pwcvqz-ri
apr-cc-pVQZ	apr-cc-pvqz	apr-cc-pvqz-jkfit	apr-cc-pvqz-ri
apr-cc-pV(Q+d)Z	apr-cc-pv_qpd_z	apr-cc-pvqz-jkfit	apr-cc-pvqz-ri
apr-cc-pCVQZ	apr-cc-pcvqz	-	-
apr-cc-pCV(Q+d)Z	apr-cc-pcv_qpd_z	-	-
apr-cc-pwCVQZ	apr-cc-pwcvqz	apr-cc-pvqz-jkfit	apr-cc-pwcvqz-ri
apr-cc-pwCV(Q+d)Z	apr-cc-pwcv_qpd_z	apr-cc-pvqz-jkfit	apr-cc-pwcvqz-ri
d-aug-cc-pVQZ	d-aug-cc-pvqz	-	-
d-aug-cc-pCVQZ	d-aug-cc-pcvqz	-	-
d-aug-cc-pwCVQZ	d-aug-cc-pwcvqz	-	-
cc-pVQZ-DK	cc-pvqz-dk	-	-
cc-pCVQZ-DK	cc-pcvqz-dk	-	-
cc-pwCVQZ-DK	cc-pwcvqz-dk	-	-
aug-cc-pVQZ-DK	aug-cc-pvqz-dk	-	-
aug-cc-pCVQZ-DK	aug-cc-pcvqz-dk	-	-
aug-cc-pwCVQZ-DK	aug-cc-pwcvqz-dk	-	-
heavy-aug-cc-pVQZ-DK	heavy-aug-cc-pvqz-dk	-	-
heavy-aug-cc-pCVQZ-DK	heavy-aug-cc-pcvqz-dk	-	-
heavy-aug-cc-pwCVQZ-DK	heavy-aug-cc-pwcvqz-dk	-	-
cc-pV5Z	cc-pv5z	cc-pv5z-jkfit	cc-pv5z-ri
cc-pV(5+d)Z	cc-pv_5pd_z	cc-pv5z-jkfit	cc-pv5z-ri
cc-pCV5Z	cc-pcv5z	-	-
cc-pCV(5+d)Z	cc-pcv_5pd_z	-	-
cc-pwCV5Z	cc-pwcv5z	cc-pv5z-jkfit	cc-pwcv5z-ri
cc-pwCV(5+d)Z	cc-pwcv_5pd_z	cc-pv5z-jkfit	cc-pwcv5z-ri

aug-cc-pV5Z	aug-cc-pv5z	aug-cc-pv5z-jkfit	aug-cc-pv5z-ri
aug-cc-pV(5+d)Z	aug-cc-pv_5pd_z	aug-cc-pv5z-jkfit	aug-cc-pv5z-ri
aug-cc-pCV5Z	aug-cc-pcv5z	-	-
aug-cc-pCV(5+d)Z	aug-cc-pcv_5pd_z	-	-
aug-cc-pwCV5Z	aug-cc-pwcv5z	aug-cc-pv5z-jkfit	aug-cc-pwcv5z-ri
aug-cc-pwCV(5+d)Z	aug-cc-pwcv_5pd_z	aug-cc-pv5z-jkfit	aug-cc-pwcv5z-ri
heavy-aug-cc-pV5Z	heavy-aug-cc-pv5z	heavy-aug-cc-pv5z-jkfit	heavy-aug-cc-pv5z-ri
heavy-aug-cc-pV(5+d)Z	heavy-aug-cc-pv_5pd_z	heavy-aug-cc-pv5z-jkfit	heavy-aug-cc-pv5z-ri
heavy-aug-cc-pCV5Z	heavy-aug-cc-pcv5z	-	-
heavy-aug-cc-pCV(5+d)Z	heavy-aug-cc-pcv_5pd_z	-	-
heavy-aug-cc-pwCV5Z	heavy-aug-cc-pwcv5z	heavy-aug-cc-pv5z-jkfit	heavy-aug-cc-pwcv5z-ri
heavy-aug-cc-pwCV(5+d)Z	heavy-aug-cc-pwcv_5pd_z	heavy-aug-cc-pv5z-jkfit	heavy-aug-cc-pwcv5z-ri
jun-cc-pV5Z	jun-cc-pv5z	jun-cc-pv5z-jkfit	jun-cc-pv5z-ri
jun-cc-pV(5+d)Z	jun-cc-pv_5pd_z	jun-cc-pv5z-jkfit	jun-cc-pv5z-ri
jun-cc-pCV5Z	jun-cc-pcv5z	-	-
jun-cc-pCV(5+d)Z	jun-cc-pcv_5pd_z	-	-
jun-cc-pwCV5Z	jun-cc-pwcv5z	jun-cc-pv5z-jkfit	jun-cc-pwcv5z-ri
jun-cc-pwCV(5+d)Z	jun-cc-pwcv_5pd_z	jun-cc-pv5z-jkfit	jun-cc-pwcv5z-ri
may-cc-pV5Z	may-cc-pv5z	may-cc-pv5z-jkfit	may-cc-pv5z-ri
may-cc-pV(5+d)Z	may-cc-pv_5pd_z	may-cc-pv5z-jkfit	may-cc-pv5z-ri
may-cc-pCV5Z	may-cc-pcv5z	-	-
may-cc-pCV(5+d)Z	may-cc-pcv_5pd_z	-	-
may-cc-pwCV5Z	may-cc-pwcv5z	may-cc-pv5z-jkfit	may-cc-pwcv5z-ri
may-cc-pwCV(5+d)Z	may-cc-pwcv_5pd_z	may-cc-pv5z-jkfit	may-cc-pwcv5z-ri
apr-cc-pV5Z	apr-cc-pv5z	apr-cc-pv5z-jkfit	apr-cc-pv5z-ri
apr-cc-pV(5+d)Z	apr-cc-pv_5pd_z	apr-cc-pv5z-jkfit	apr-cc-pv5z-ri
apr-cc-pCV5Z	apr-cc-pcv5z	-	-
apr-cc-pCV(5+d)Z	apr-cc-pcv_5pd_z	-	-
apr-cc-pwCV5Z	apr-cc-pwcv5z	apr-cc-pv5z-jkfit	apr-cc-pwcv5z-ri
apr-cc-pwCV(5+d)Z	apr-cc-pwcv_5pd_z	apr-cc-pv5z-jkfit	apr-cc-pwcv5z-ri
mar-cc-pV5Z	mar-cc-pv5z	mar-cc-pv5z-jkfit	mar-cc-pv5z-ri
mar-cc-pV(5+d)Z	mar-cc-pv_5pd_z	mar-cc-pv5z-jkfit	mar-cc-pv5z-ri
mar-cc-pCV5Z	mar-cc-pcv5z	-	-
mar-cc-pCV(5+d)Z	mar-cc-pcv_5pd_z	-	-
mar-cc-pwCV5Z	mar-cc-pwcv5z	mar-cc-pv5z-jkfit	mar-cc-pwcv5z-ri
mar-cc-pwCV(5+d)Z	mar-cc-pwcv_5pd_z	mar-cc-pv5z-jkfit	mar-cc-pwcv5z-ri
d-aug-cc-pV5Z	d-aug-cc-pv5z	-	-
d-aug-cc-pCV5Z	d-aug-cc-pcv5z	-	-
d-aug-cc-pwCV5Z	d-aug-cc-pwcv5z	-	-
cc-pV5Z-DK	cc-pv5z-dk	-	-
cc-pCV5Z-DK	cc-pcv5z-dk	-	-
cc-pwCV5Z-DK	cc-pwcv5z-dk	-	-
aug-cc-pV5Z-DK	aug-cc-pv5z-dk	-	-
aug-cc-pCV5Z-DK	aug-cc-pcv5z-dk	-	-
aug-cc-pwCV5Z-DK	aug-cc-pwcv5z-dk	-	-
heavy-aug-cc-pV5Z-DK	heavy-aug-cc-pv5z-dk	-	-
heavy-aug-cc-pCV5Z-DK	heavy-aug-cc-pcv5z-dk	-	-

heavy-aug-cc-pwCV5Z-DK	heavy-aug-cc-pwcv5z-dk	-	-
cc-pV6Z	cc-pv6z	-	cc-pv6z-ri
cc-pV(6+d)Z	cc-pv_6pd_z	-	cc-pv6z-ri
cc-pCV6Z	cc-pcv6z	-	-
cc-pCV(6+d)Z	cc-pcv_6pd_z	-	-
aug-cc-pV6Z	aug-cc-pv6z	-	aug-cc-pv6z-ri
aug-cc-pV(6+d)Z	aug-cc-pv_6pd_z	-	aug-cc-pv6z-ri
aug-cc-pCV6Z	aug-cc-pcv6z	-	-
aug-cc-pCV(6+d)Z	aug-cc-pcv_6pd_z	-	-
heavy-aug-cc-pV6Z	heavy-aug-cc-pv6z	-	heavy-aug-cc-pv6z-ri
heavy-aug-cc-pV(6+d)Z	heavy-aug-cc-pv_6pd_z	-	heavy-aug-cc-pv6z-ri
heavy-aug-cc-pCV6Z	heavy-aug-cc-pcv6z	-	-
heavy-aug-cc-pCV(6+d)Z	heavy-aug-cc-pcv_6pd_z	-	-
jun-cc-pV6Z	jun-cc-pv6z	-	jun-cc-pv6z-ri
jun-cc-pV(6+d)Z	jun-cc-pv_6pd_z	-	jun-cc-pv6z-ri
jun-cc-pCV6Z	jun-cc-pcv6z	-	-
jun-cc-pCV(6+d)Z	jun-cc-pcv_6pd_z	-	-
may-cc-pV6Z	may-cc-pv6z	-	may-cc-pv6z-ri
may-cc-pV(6+d)Z	may-cc-pv_6pd_z	-	may-cc-pv6z-ri
may-cc-pCV6Z	may-cc-pcv6z	-	-
may-cc-pCV(6+d)Z	may-cc-pcv_6pd_z	-	-
apr-cc-pV6Z	apr-cc-pv6z	-	apr-cc-pv6z-ri
apr-cc-pV(6+d)Z	apr-cc-pv_6pd_z	-	apr-cc-pv6z-ri
apr-cc-pCV6Z	apr-cc-pcv6z	-	-
apr-cc-pCV(6+d)Z	apr-cc-pcv_6pd_z	-	-
mar-cc-pV6Z	mar-cc-pv6z	-	mar-cc-pv6z-ri
mar-cc-pV(6+d)Z	mar-cc-pv_6pd_z	-	mar-cc-pv6z-ri
mar-cc-pCV6Z	mar-cc-pcv6z	-	-
mar-cc-pCV(6+d)Z	mar-cc-pcv_6pd_z	-	-
feb-cc-pV6Z	feb-cc-pv6z	-	feb-cc-pv6z-ri
feb-cc-pV(6+d)Z	feb-cc-pv_6pd_z	-	feb-cc-pv6z-ri
feb-cc-pCV6Z	feb-cc-pcv6z	-	-
feb-cc-pCV(6+d)Z	feb-cc-pcv_6pd_z	-	-
d-aug-cc-pV6Z	d-aug-cc-pv6z	-	-
d-aug-cc-pCV6Z	d-aug-cc-pcv6z	-	-
STO-3G	sto-3g	def2-universal-jkfit	def2-svp-ri
STO-6G	sto-6g	def2-universal-jkfit	def2-svp-ri
3-21G	3-21g	def2-universal-jkfit	def2-svp-ri
6-31G	6-31g	cc-pvdz-jkfit	cc-pvdz-ri
6-31G(d)	6-31g_d_	cc-pvdz-jkfit	cc-pvdz-ri
6-31G(d,p)	6-31g_d_p_	cc-pvdz-jkfit	cc-pvdz-ri
6-31G*	6-31g_d_	cc-pvdz-jkfit	cc-pvdz-ri
6-31G**	6-31g_d_p_	cc-pvdz-jkfit	cc-pvdz-ri
6-31+G	6-31pg	heavy-aug-cc-pvdz-jkfit	heavy-aug-cc-pvdz-ri
6-31+G(d)	6-31pg_d_	heavy-aug-cc-pvdz-jkfit	heavy-aug-cc-pvdz-ri
6-31+G(d,p)	6-31pg_d_p_	heavy-aug-cc-pvdz-jkfit	heavy-aug-cc-pvdz-ri
6-31+G*	6-31pg_d_	heavy-aug-cc-pvdz-jkfit	heavy-aug-cc-pvdz-ri

6-31+G**	6-31pg_d_p_	heavy-aug-cc-pvdz-jkfit	heavy-aug-cc-pvdz-ri
6-31++G	6-31ppg	aug-cc-pvdz-jkfit	aug-cc-pvdz-ri
6-31++G(d)	6-31ppg_d_	aug-cc-pvdz-jkfit	aug-cc-pvdz-ri
6-31++G(d,p)	6-31ppg_d_p_	aug-cc-pvdz-jkfit	aug-cc-pvdz-ri
6-31++G*	6-31ppg_d_	aug-cc-pvdz-jkfit	aug-cc-pvdz-ri
6-31++G**	6-31ppg_d_p_	aug-cc-pvdz-jkfit	aug-cc-pvdz-ri
6-311G	6-311g	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(d)	6-311g_d_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(d,p)	6-311g_d_p_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G*	6-311g_d_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G**	6-311g_d_p_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(2d)	6-311g_2d_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(2d,p)	6-311g_2d_p_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(2d,2p)	6-311g_2d_2p_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(2df)	6-311g_2df_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(2df,p)	6-311g_2df_p_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(2df,2p)	6-311g_2df_2p_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(2df,2pd)	6-311g_2df_2pd_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(3df)	6-311g_3df_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(3df,p)	6-311g_3df_p_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(3df,2p)	6-311g_3df_2p_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(3df,2pd)	6-311g_3df_2pd_	cc-pvtz-jkfit	cc-pvtz-ri
6-311G(3df,3pd)	6-311g_3df_3pd_	cc-pvtz-jkfit	cc-pvtz-ri
6-311+G	6-311pg	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(d)	6-311pg_d_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(d,p)	6-311pg_d_p_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G*	6-311pg_d_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G**	6-311pg_d_p_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(2d)	6-311pg_2d_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(2d,p)	6-311pg_2d_p_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(2d,2p)	6-311pg_2d_2p_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(2df)	6-311pg_2df_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(2df,p)	6-311pg_2df_p_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(2df,2p)	6-311pg_2df_2p_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(2df,2pd)	6-311pg_2df_2pd_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(3df)	6-311pg_3df_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(3df,p)	6-311pg_3df_p_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(3df,2p)	6-311pg_3df_2p_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri

6-311+G(3df,2pd)	6-311pg_3df_2pd_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311+G(3df,3pd)	6-311pg_3df_3pd_	heavy-aug-cc-pvtz-jkfit	heavy-aug-cc-pvtz-ri
6-311++G	6-311ppg	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(d)	6-311ppg_d_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(d,p)	6-311ppg_d_p_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G*	6-311ppg_d_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G**	6-311ppg_d_p_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(2d)	6-311ppg_2d_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(2d,p)	6-311ppg_2d_p_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(2d,2p)	6-311ppg_2d_2p_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(2df)	6-311ppg_2df_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(2df,p)	6-311ppg_2df_p_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(2df,2p)	6-311ppg_2df_2p_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(2df,2pd)	6-311ppg_2df_2pd_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(3df)	6-311ppg_3df_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(3df,p)	6-311ppg_3df_p_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(3df,2p)	6-311ppg_3df_2p_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(3df,2pd)	6-311ppg_3df_2pd_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
6-311++G(3df,3pd)	6-311ppg_3df_3pd_	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
def2-SV(P)	def2-sv_p_	def2-universal-jkfit	def2-sv_p_-ri
def2-mSVP	def2-msvp	def2-universal-jkfit	def2-svp-ri
def2-SVP	def2-svp	def2-universal-jkfit	def2-svp-ri
def2-SVPD	def2-svpd	def2-universal-jkfit	def2-svpd-ri
def2-TZVP	def2-tzvp	def2-universal-jkfit	def2-tzvp-ri
def2-TZVPD	def2-tzvpd	def2-universal-jkfit	def2-tzvpd-ri
def2-TZVPP	def2-tzvpp	def2-universal-jkfit	def2-tzvpp-ri
def2-TZVPPD	def2-tzvppd	def2-universal-jkfit	def2-tzvppd-ri
def2-QZVP	def2-qzvp	def2-universal-jkfit	def2-qzvp-ri
def2-QZVPD	def2-qzvpd	def2-universal-jkfit	-
def2-QZVPP	def2-qzvpp	def2-universal-jkfit	def2-qzvpp-ri
def2-QZVPPD	def2-qzvppd	def2-universal-jkfit	def2-qzvppd-ri
aug-pcseg-0	aug-pcseg-0	def2-universal-jkfit	def2-sv_p_-ri
aug-pcseg-1	aug-pcseg-1	def2-universal-jkfit	def2-svpd-ri
aug-pcseg-2	aug-pcseg-2	def2-universal-jkfit	def2-tzvppd-ri
aug-pcseg-3	aug-pcseg-3	def2-universal-jkfit	def2-qzvppd-ri
aug-pcseg-4	aug-pcseg-4	aug-cc-pv5z-jkfit	aug-cc-pv5z-ri
aug-pcSseg-0	aug-pcsseg-0	def2-universal-jkfit	def2-sv_p_-ri
aug-pcSseg-1	aug-pcsseg-1	def2-universal-jkfit	def2-svpd-ri
aug-pcSseg-2	aug-pcsseg-2	def2-universal-jkfit	def2-tzvppd-ri
aug-pcSseg-3	aug-pcsseg-3	def2-universal-jkfit	def2-qzvppd-ri
aug-pcSseg-4	aug-pcsseg-4	aug-cc-pv5z-jkfit	aug-cc-pwcv5z-ri
pcseg-0	pcseg-0	def2-universal-jkfit	def2-sv_p_-ri
pcseg-1	pcseg-1	def2-universal-jkfit	def2-svp-ri
pcseg-2	pcseg-2	def2-universal-jkfit	def2-tzvpp-ri
pcseg-3	pcseg-3	def2-universal-jkfit	def2-qzvpp-ri
pcseg-4	pcseg-4	cc-pv5z-jkfit	cc-pv5z-ri
pcSseg-0	pcsseg-0	def2-universal-jkfit	def2-sv_p_-ri
pcSseg-1	pcsseg-1	def2-universal-jkfit	def2-svp-ri
pcSseg-2	pcsseg-2	def2-universal-jkfit	def2-tzvpp-ri
pcSseg-3	pcsseg-3	def2-universal-jkfit	def2-qzvpp-ri
pcSseg-4	pcsseg-4	cc-pv5z-jkfit	cc-pwcv5z-ri
minix	minix	def2-universal-jkfit	def2-svp-ri

DZ	dz	-	-
DZP	dzp	-	-
DZVP	dzvp	dgauss-dzvp-mix	dgauss-dzvp-autoaux
psi3-DZP	psi3-dzp	-	-
psi3-TZ2P	psi3-tz2p	-	-
psi3-TZ2PF	psi3-tz2pf	-	-
sadlej-lpol-dl	sadlej-lpol-dl	-	-
sadlej-lpol-ds	sadlej-lpol-ds	-	-
sadlej-lpol-fl	sadlej-lpol-fl	-	-
sadlej-lpol-fs	sadlej-lpol-fs	-	-
2zapa-nr	2zapa-nr	aug-cc-pvtz-jkfit	aug-cc-pvtz-ri
3zapa-nr	3zapa-nr	aug-cc-pvqz-jkfit	aug-cc-pvqz-ri
4zapa-nr	4zapa-nr	aug-cc-pv5z-jkfit	aug-cc-pv5z-ri
5zapa-nr	5zapa-nr	aug-cc-pv5z-jkfit	aug-cc-pv6z-ri
6zapa-nr	6zapa-nr	aug-cc-pv6z-ri	aug-cc-pv6z-ri
7zapa-nr	7zapa-nr	aug-cc-pv6z-ri	aug-cc-pv6z-ri
cc-pvqz-f12	cc-pvqz-f12	cc-pv5z-jkfit	cc-pv5z-ri
cc-pvtz-f12	cc-pvtz-f12	cc-pvqz-jkfit	cc-pvqz-ri
cc-pvqz-f12	cc-pvqz-f12	cc-pv5z-jkfit	cc-pv5z-ri

7.3 Assigning Basis Sets for Individual Atoms

Different basis sets can be assigned to individual atoms in the `BASIS`, `AUX_BASIS_SCF`, `AUX_BASIS_CORR`, and `SAD_AUX_BASIS` blocks. For example:

```
$set_molecule
  0 1
  C     -1.948844    0.619359   -0.000000
  C     -0.016018    0.608680    0.000000
  H      0.336565   -0.131599   -0.687440
  H      0.345972    1.571195   -0.295683
  H      0.339394    0.380532    0.983124
  H     -2.301427    1.359637    0.687439
  H     -2.304256    0.847506   -0.983124
  H     -2.310834   -0.343156    0.295683
  Zn    -0.186875    3.128829    0.000000
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method mp2
  reference uhf
  jobtype energy
$end_globals
```

```
$set_basis
  Zn def2-qzvp
$end_basis

$set_aux_basis_scf
  Zn def2-universal-jkfit
$end_aux_basis_scf

$set_aux_basis_corr
  Zn def2-qzvp-ri
$end_aux_basis_corr
```

The relevant part of the output file for this example:

```
 CC-PVDZ primary basis set file will be read for the C atom...
 CC-PVDZ primary basis set file will be read for the C atom...
 CC-PVDZ primary basis set file will be read for the H atom...
 CC-PVDZ primary basis set file will be read for the H atom...
 CC-PVDZ primary basis set file will be read for the H atom...
 CC-PVDZ primary basis set file will be read for the H atom...
 CC-PVDZ primary basis set file will be read for the H atom...
 CC-PVDZ primary basis set file will be read for the H atom...
 DEF2-QZVP primary basis set file will be read for the ZN atom...
 Nmaxthread: 8
 Nthread:  1
 Nshell: 56
 Lmax:  4
 Npmax: 11
 Nao: 142
 Ndocc: 24
 Nsocc:  0
 NoccA: 24
 NoccB: 24
 Primary basis set info has been read.

 CC-PVDZ-JKFIT aux basis set file will be read for the C atom...
 CC-PVDZ-JKFIT aux basis set file will be read for the C atom...
 CC-PVDZ-JKFIT aux basis set file will be read for the H atom...
 CC-PVDZ-JKFIT aux basis set file will be read for the H atom...
 CC-PVDZ-JKFIT aux basis set file will be read for the H atom...
 CC-PVDZ-JKFIT aux basis set file will be read for the H atom...
 CC-PVDZ-JKFIT aux basis set file will be read for the H atom...
 CC-PVDZ-JKFIT aux basis set file will be read for the H atom...
 DEF2-UNIVERSAL-JKFIT aux basis set file will be read for the ZN atom...
 Nshell_aux: 102
 Naux: 278
 Aux basis set info has been read.
```

⋮

```
 CC-PVDZ primary basis set file will be read for the C atom...
 CC-PVDZ primary basis set file will be read for the C atom...
```

```
CC-PVDZ primary basis set file will be read for the H atom...
CC-PVDZ primary basis set file will be read for the H atom...
CC-PVDZ primary basis set file will be read for the H atom...
CC-PVDZ primary basis set file will be read for the H atom...
CC-PVDZ primary basis set file will be read for the H atom...
CC-PVDZ primary basis set file will be read for the H atom...
DEF2-QZVP primary basis set file will be read for the ZN atom...
Nmaxthread: 8
Nthread:  1
Nshell: 56
Lmax:  4
Npmax: 11
Nao: 142
Ndocc: 24
Nsocc:  0
NoccA: 24
NoccB: 24
Primary basis set info has been read.

CC-PVDZ-RI aux basis set file will be read for the C atom...
CC-PVDZ-RI aux basis set file will be read for the C atom...
CC-PVDZ-RI aux basis set file will be read for the H atom...
CC-PVDZ-RI aux basis set file will be read for the H atom...
CC-PVDZ-RI aux basis set file will be read for the H atom...
CC-PVDZ-RI aux basis set file will be read for the H atom...
CC-PVDZ-RI aux basis set file will be read for the H atom...
CC-PVDZ-RI aux basis set file will be read for the H atom...
DEF2-QZVP-RI aux basis set file will be read for the ZN atom...
```

7.4 User-Defined Basis Sets

`MacroQC` uses Gaussian format for basis sets. For example, the `cc-pvdz` basis set for the H atom:

```
H     0
S   3   1.00
     13.0100000              0.0196850
      1.9620000              0.1379770
      0.4446000              0.4781480
S   1   1.00
      0.1220000              1.0000000
P   1   1.00
      0.7270000              1.0000000
****
```

To submit a general use-defined basis to `MacroQC` one needs to prepare a `.gbs` file in which the basis set should be given in the Gaussian format. For example, if one prepares a basis set file titled `mybasis.gbs`, this file should be placed in the `MQCIDIR/macroqc/share/basis` directory. `MQCIDIR` is the directory where `MacroQC` is installed. For example: `/opt/macroqc`.

Many basis sets can be obtained from the *Basis Set Exchange* [26] website: Basis Set Exchange

7.5 Integral Library Options

- **PRINT_LEVEL**
  The amount of information to print to the output file.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **MAX_AM**
  Maximum angular momentum value for the primary basis set.
  TYPE: Integer
  DEFAULT: 7
  OPTIONS: n User-defined value.

- **MAX_AM_AUX**
  Maximum angular momentum value for the auxiliary basis set.
  TYPE: Integer
  DEFAULT: 7
  OPTIONS: n User-defined value.

- **TAYLOR_ORDER**
  Order of Taylor series expansion to compute Boys function.
  TYPE: Integer
  DEFAULT: 6
  OPTIONS: n User-defined value.

- **BOYS_MAXITER**
  Maximum number of iterations for the Boys grid.
  TYPE: Integer
  DEFAULT: 10000
  OPTIONS: n User-defined value.

- **X_MAX**
  The maximum value of x in the Boys grid.
  TYPE: Double
  DEFAULT: 30.0
  OPTIONS: n User-defined value.

- **INTEGRAL_CUTOFF**
  The cutoff value for numerical procedures.
  TYPE: Double
  DEFAULT: 1.0e-10
  OPTIONS: n User-defined value.

- **BOYS_CUTOFF**
  The cutoff value for numerical procedures in the Boys grid.
  TYPE: Double
  DEFAULT: 1.0E-15
  OPTIONS: n User-defined value.

- **BOYS_ZERO**
  The threshold value for numerical procedures in the Boys grid.
  TYPE: Double
  DEFAULT: 1.0E-6
  OPTIONS: n User-defined value.

- **DELTA_X**
  Step size in the Boys grid.
  TYPE: Double
  DEFAULT: 0.1
  OPTIONS: n User-defined value.

- **3INDEX_ALGORITHM**
  The algorithm used for computations of 3-index ERIs.
  TYPE: Sring
  DEFAULT: AUTO
  OPTIONS: AUTO, OS1, OS2, MD4

- **BOYS_METHOD**
  The algorithm is used to compute grid values.
  TYPE: Sring
  DEFAULT: GHP
  OPTIONS: GHP, SAUNDERS

8 SCF : Self-Consistent Field

Code Authors: U. Bozkaya and Y. Alagöz.

8.1 Overview

`MacroQC` performs self-consistent field computations for restricted, unrestricted, and restricted open-shell Hartree–Fock (RHF, UHF, and ROHF) references. `MacroQC` provides several different algorithms to obtain the HF solution, such as the conventional Roothan-Hall procedure, approximately second-order algorithm (ASO-SCF) [27], and the quadratically convergent SCF algorithm (QC-SCF). The `scf` module employs several initial guesses, such as the generalized Wolfsberg-Helmholtz (GWH) [28], superposition of atomic density (SAD), and the core Hamiltonian. If there is enough memory, `MacroQC` uses an incore JK-algorithm to build the Fock matrix. In the case of limited memory, one may employ an integral-direct approach [20].

8.2 ROHF

The ROHF Fock matrix can be written as follows:

$$\boldsymbol{F} \;=\; \begin{pmatrix} F_{cc} & F_{co} & F_{cv} \\ F_{oc} & F_{oo} & F_{ov} \\ F_{vc} & F_{vo} & F_{vv} \end{pmatrix} \tag{4}$$

where c, o, v are closed, open, and virtual parts. In general, the ROHF effective Fock matrix can be written as follows:

$$\boldsymbol{F} \;=\; \begin{bmatrix} A_{cc}F^{\alpha} + B_{cc}F^{\beta} & F^{\beta} & \frac{1}{2}\big(F^{\alpha} + F^{\beta}\big) \\ F^{\beta} & A_{oo}F^{\alpha} + B_{oo}F^{\beta} & F^{\alpha} \\ \frac{1}{2}\big(F^{\alpha} + F^{\beta}\big) & F^{\alpha} & A_{vv}F^{\alpha} + B_{vv}F^{\beta} \end{bmatrix} \tag{5}$$

In `MacroQC` there are several options available for ROHF coefficients (Table 6), such as Guest and Saunders [29], Roothan [30], Davidson [31], Binkley, Pople, Dobosh [32], McWeeny and Diercksen [33], Faegri and Manne [34], Plakhutin, Gorelik, and Breslavskaya [35], and GAMESS GVB program [36].

Table 6: ROHF coefficients available in `MacroQC`.

	Acc	***Bcc***	***Aoo***	***Boo***	***Avv***	***Bvv***
Guest and Saunders	1/2	1/2	1/2	1/2	1/2	1/2
Roothaan	−1/2	3/2	1/2	1/2	3/2	−1/2
Davidson	1/2	1/2	1	0	1	0
Binkley, Pople, Dobosh	1/2	1/2	1	0	0	1
McWeeny and Diercksen	1/3	2/3	1/3	1/3	2/3	1/3
Faegri and Manne	1/2	1/2	1	0	1/2	1/2
Plakhutin, Gorelik, and Breslavskaya	0	1	1	0	1	0
GAMESS GVB program	1/2	1/2	1/2	0	1/2	1/2
Pure Alpha	1	0	1	0	1	0
Pure Beta	0	1	0	1	0	1

8.2.1 The ROHF gradient

The ROHF energy is the same functional of the two spin densities as the UHF one — ROHF only constrains the two sets of orbitals to be common — so every integral-derivative term is shared, and the constraint enters in one place, the energy-weighted density. Both are stationary functionals of orbitals subject to $\boldsymbol{C}^T\boldsymbol{S}\boldsymbol{C} = \mathbf{1}$, so the Lagrange multiplier is

$$\boldsymbol{\Lambda} \;=\; \mathrm{sym}\Big(\tilde{\boldsymbol{F}}^{\alpha}\boldsymbol{n}^{\alpha} \;+\; \tilde{\boldsymbol{F}}^{\beta}\boldsymbol{n}^{\beta}\Big), \qquad \tilde{\boldsymbol{F}}^{\sigma} \;=\; \boldsymbol{C}^T\boldsymbol{F}^{\sigma}\boldsymbol{C}, \tag{6}$$

with $\boldsymbol{n}^\sigma$ the occupation numbers of spin σ, and $\boldsymbol{W} = -\boldsymbol{C}\boldsymbol{\Lambda}\boldsymbol{C}^T$. Since $\tilde{\boldsymbol{F}}^\sigma$ is symmetric this is $\tilde{\boldsymbol{F}}^\sigma$ weighted by how many of its two indices are occupied in that spin: one if both, one half if one, zero if neither.

For RHF and UHF the orbitals are canonical, $\tilde{\boldsymbol{F}}$ is diagonal in the occupied block, the occupied-virtual block vanishes by Brillouin's theorem, and this collapses to the familiar sum over orbital energies. For ROHF it does not: the alpha and beta Fock matrices are not diagonal in the ROHF orbitals — only their ROHF combination is — and the Brillouin conditions hold for that combination rather than for the two matrices separately, so the whole matrix is needed, occupied-virtual block included. No coupled-perturbed equation is involved: the SCF energy is variational, and the multiplier is all the orbital response there is.

8.3 Quasi-Restricted Orbitals

Setting `QRO` to true in the `$set_scf` block replaces the converged unrestricted orbitals by quasi-restricted ones [F. Neese, *J. Am. Chem. Soc.* **128**, 10213 (2006)]. The construction follows the recipe as stated by Hansen, Liakos and Neese [*J. Chem. Phys.* **135**, 214102 (2011)]:

1. the spin-summed density $D = D^\alpha + D^\beta$ is diagonalized to give the unrestricted natural orbitals and their occupation numbers, sorted downwards;
2. the leading N_β of them are taken as doubly occupied (DOMO), the next $N_\alpha - N_\beta$ as singly occupied (SOMO) and the rest as virtual (VMO);
3. each block is canonicalized with its own Fock operator: F^β for the DOMOs, $\frac{1}{2}(F^\alpha + F^\beta)$ for the SOMOs and F^α for the VMOs.

Both spins then share one set of spatial orbitals, so the determinant is a spin eigenfunction. The module prints $\langle S^2 \rangle$ before and after as a check: for the methyl radical in cc-pVDZ it goes from 0.761162 to exactly 0.750000, and for triplet O_2 from 2.033025 to 2.000000. For a closed-shell system the QROs reduce to the canonical restricted orbitals.

The SCF energy is not affected, since this is a transformation applied after convergence. QROs are the reference of choice for the open-shell local pair-natural orbital methods, which need a single set of spatial orbitals; they are also useful whenever spin contamination of the UHF determinant would otherwise be carried into a correlated treatment.

8.4 Input Examples

*Single-point energy computation at the `rhf/cc-pvdz` level:

```
$set_molecule H2O
    0 1
    O    0.00000    0.00000   -0.06577
```

```
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    method scf
    reference rhf
    jobtype energy
$end_globals
```

*Single-point energy computation for a molecule in an electric field.

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    method scf
    reference rhf
    jobtype energy
    electric_field xyz
    electric_strength [0.1;0.1;0.1]
$end_globals
```

8.5 SCF Module Options

- **SCF_TYPE**
  Source of the two-electron integrals used to build the Fock matrix. This single option replaces the earlier `do_df`, `eri` and `cholesky` keywords, of which only certain combinations were meaningful.
  TYPE: String
  DEFAULT: DF_INCORE
  OPTIONS: DF_INCORE, DF_DIRECT, CONV_ERI_INCORE, CONV_ERI_DIRECT, CD

  - DF_INCORE: Density fitting with the three-index tensor $B(Q|\mu\nu)$ built once and kept.
  - DF_DIRECT: Density fitting with the three-index integrals rebuilt on the fly, so the full $B(Q|\mu\nu)$ never resides in memory.
  - CONV_ERI_INCORE: Conventional four-index integrals, computed once and stored in packed form ($\frac{1}{2}M(M+1)$ numbers with $M = \frac{1}{2}n_{\mathrm{bf}}(n_{\mathrm{bf}}+1)$), after which each Fock build is a sweep over that array.
  - CONV_ERI_DIRECT: Conventional four-index integrals generated on the fly by the Obara–Saika kernel and discarded after use. No auxiliary basis is involved and nothing of size n_{bf}^4 is stored, so the energy carries no fitting error. Available for Hartree–Fock and for Kohn–Sham DFT with every functional, including the range-separated hybrids, whose long-range exchange is obtained from exact $\mathrm{erf}(\omega r_{12})/r_{12}$ integrals rather than a fitted or semi-numerical substitute.
  - CD: Cholesky decomposition of the ERI tensor, $(\mu\nu|\lambda\sigma) \approx \sum_Q L_Q(\mu\nu)L_Q(\lambda\sigma)$. The factors play exactly the role of the DF $B(Q|\mu\nu)$, so the J/K builders are unchanged; no auxiliary basis and no $J^{-1/2}$ are involved, and the accuracy is set by a single threshold instead of by the quality of a fitting basis. Implemented for Hartree–Fock only (RHF/UHF/ROHF) and for ENERGIES only: DFT, analytic gradients and COSX require DF. The factors are built in core.

  The conventional four-index routes always use the OS2 (Obara–Saika) kernel, irrespective of **ERI_ALGORITHM** in the INTS block: it is the fastest of the four-index kernels and the only one carrying the erf-attenuated integrals that range-separated functionals need.

  Not every choice covers every method. CONV_ERI_DIRECT is the only one that spans all of Kohn–Sham DFT, range-separated hybrids included:

SCF_TYPE	Hartree–Fock	LDA/GGA/meta-GGA	global hybrid	range-separated
DF_INCORE	yes	yes	yes	yes
DF_DIRECT	yes	yes	yes	only with COSX
CONV_ERI_INCORE	yes	yes	yes	no
CONV_ERI_DIRECT	yes	yes	yes	yes
CD	yes	no	no	no

The two gaps have the same cause seen from opposite sides. DF_DIRECT never forms the erf-attenuated $B_\omega(Q|\mu\nu)$, which only the incore fit builds, so the long-range exchange has to come from the COSX grid instead. CONV_ERI_INCORE stores one packed array, and it holds the bare integrals; contracting the long-range term against it would silently return standard exchange, an error of order 1 E_h rather than a small one, so that combination is refused. CONV_ERI_DIRECT has neither problem: it simply generates a second set of quartets over $\mathrm{erf}(\omega r_{12})/r_{12}$. CD is restricted to Hartree–Fock for reasons unrelated to the above (see the CD entry).

- **CHOLESKY_TOLERANCE**
  Threshold of the CD in the SCF. The decomposition stops when the largest remaining diagonal of the residual matrix falls below this value, which also bounds every reconstructed integral. If the post-SCF module (DFOCC) is run with CD at the SAME threshold, it reuses these factors instead of decomposing again – the factors are AO-basis quantities and do not depend on the orbitals. With a different threshold, or with DF in the post-SCF step, the post-SCF side builds what it needs from scratch.
  TYPE: Double
  DEFAULT: 1.0e-4
  OPTIONS: n User-defined tolerance value.

- **CHOLESKY_SPAN**
  Pivot acceptance factor inside one Cholesky block: candidates are taken while their residual diagonal exceeds `span` times the largest one. The decomposition is valid for any pivot order, so this only trades the number of vectors against speed; 1.0 selects strict-greedy pivoting.
  TYPE: Double
  DEFAULT: 1.0e-2
  OPTIONS: n User-defined value in $(0, 1]$.

- **ELECTRIC_FIELD**
  Applies an electric field in the given direction. The electric field strength, which will be applied, is controlled by the EF_STRENGTH option.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, X, Y, Z, XY, XZ, YZ, XYZ

  - NONE: It does not apply any electric field.
  - X: Applies an electric field in the x-direction.
  - Y: Applies an electric field in the y-direction.
  - Z: Applies an electric field in the z-direction.
  - XY: Applies an electric field in the x and y directions.
  - XZ: Applies an electric field in the x and z directions.
  - YZ: Applies an electric field in the y and z directions.
  - XYZ: Applies an electric field in the x, y, and z directions.

- **EF_STRENGTH**
  The strength of the electric field applied in the x, y, and z directions, respectively.
  TYPE: Double Array
  DEFAULT: [0.1; 0.1; 0.1]
  OPTIONS: n User defined value.

- **DIIS_DAMP**
  Scale diagonal elements of matrix B by $1 + d$. It helps to speed up SCF converge in problematic cases, especially in open-shell systems. The damping parameter is controlled by the DAMP_PARAM option.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **DAMP_PARAM**
  The damping parameter d.
  TYPE: Double
  DEFAULT: 0.02
  OPTIONS: n User defined value.

- **DIE_IF_NOT_CONVERGED**
  Die if SCF is not converged?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **DIIS**
  Do apply DIIS for SCF iterations?
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **FOCK_DAMP**
  Do apply Fock damping during SCF iterations? The extent of damping is controlled by the **FOCK_DAMP_PERC** option.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **FOCK_DAMP_PERC**
  The percentage of damping to apply to the early Fock updates. 100 will completely stall the update, while 0 will result in a full update. A value of 20, which corresponds to 20% of the previous iteration's Fock matrix being mixed into the current Fock. Fock

damping may help to solve problems with oscillatory convergence.
TYPE: Double
DEFAULT: 20.0
OPTIONS: 0-100

- **GUESS_MIX**
  Do mix HOMO and LUMO? The GUESS_MIX option may be used for singlet molecules when UHF reference is employed.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **GUESS_PREVIOUS**
  Do use orbitals of previous geometry? This option is used for geometry optimization, where SCF orbitals guess is obtained from the previous geometry.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **LOCAL**
  Do localize occupied molecular orbitals after SCF convergence?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **LOCAL_NEWTON**
  Do polish the Foster-Boys localization with second-order steps after the Jacobi sweeps? The sweeps stop on the change of the metric and leave the orbitals a small rotation short of a stationary point of the criterion, and that distance depends on where they stopped, i.e. on the machine; a truncated local method feels it at first order in its energy (about 10^{-9} E_h) and in its analytic gradient (about 10^{-7} E_h/a_0, since the gradient assumes the localization is stationary). Worse, the sweeps can stop at a saddle point of the criterion: C_2H_5Br in STO-3G is one, and its LPNO-CCSD energy at the saddle was 1.3×10^{-4} E_h off the maximum's. The polish follows Pinski's Algorithm 7: rational-function (augmented-Hessian) steps first, which climb away from a saddle along its negative mode and bound the step along a soft one, then, once no negative mode is left and the step is small, Newton steps with a level shift (`LOCAL_NEWTON_SHIFT`) solved by preconditioned conjugate gradients. The Hessian is never stored: it acts through three matrix products per vector; the augmented matrix is diagonalized explicitly while it is smaller than 2000×2000 and by Davidson beyond. The polish reaches `LOCAL_GRAD_CONVERGENCE` in three or four steps from the sweeps' point. Foster-Boys only; Pipek-Mezey keeps its sweeps.
  TYPE: Boolean

DEFAULT: TRUE
OPTIONS: TRUE, FALSE

- **SAD_FRAC_OCC**
  Do force an even distribution of occupations across the last partially occupied orbital shell?
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **SAD_SPIN_AVERAGE**
  Do use spin-averaged occupations instead of atomic ground spin state in fractional SAD?
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **SAD_DIIS**
  Do apply DIIS for atomic UHF density computation iterations in SAD?
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **CUTOFF**
  The tolerance value used for numerical procedures.
  TYPE: Double
  DEFAULT: 1.0e-10
  OPTIONS: n User defined tolerance value.

- **E_CONVERGENCE**
  Convergence criterion for the energy.
  TYPE: Double
  DEFAULT: 1.0e-8
  OPTIONS: n User-defined tolerance value.

- **D_CONVERGENCE**
  Convergence criterion for the density matrix.
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined tolerance value.

- **QC_GUESS_E_CONVERGENCE**
  Convergence criterion for the energy for the QC-SCF guess.

TYPE: Double
DEFAULT: 1.0e-4
OPTIONS: n User-defined tolerance value.

- **QC_GUESS_D_CONVERGENCE**
  Convergence criterion for the density matrix the QC-SCF guess.
  TYPE: Double
  DEFAULT: 5.0e-3
  OPTIONS: n User-defined tolerance value.

- **MIN_OVERLAP_EIGVAL**
  Threshold value to eliminate low eigenvalues of the overlap matrix.
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined tolerance value.

- **LOCAL_CONVERGENCE**
  Convergence criterion of the Jacobi sweeps of the orbital localization: the relative change of the localization metric between two sweeps. The sweeps converge linearly and leave the orbitals about $\sqrt{\text{threshold}}$ from the maximum; with `LOCAL_NEWTON` (the default) the Newton polish that follows takes them the rest of the way, so the sweeps need not be tight.
  TYPE: Double
  DEFAULT: 1.0e-8
  OPTIONS: n User-defined tolerance value.

- **LOCAL_GRAD_CONVERGENCE**
  Convergence criterion of the second-order polish of the Foster-Boys criterion (`LOCAL_NEWTON`): the largest element of the gradient of the criterion with respect to an orbital rotation, $|\partial L/\partial \kappa_{ki}|_{\max}$ with $L = \sum_i |\langle i|\mathbf{r}|i\rangle|^2$ in bohr^2, taken outside the soft modes (those with a curvature below `LOCAL_NEWTON_SHIFT`, which the criterion cannot tell apart). The polish stops earlier, without complaint, when a step no longer lowers the gradient, which is the rounding level of the criterion (about 10^{-14} to 10^{-16} on small molecules).
  TYPE: Double
  DEFAULT: 1.0e-12
  OPTIONS: n User-defined tolerance value.

- **LOCAL_NEWTON_SHIFT**
  Level shift of the Newton stage of the polish, in bohr^2, the units of the Hessian of the criterion. A localization can be continuously degenerate, or as good as (core shells, lone pairs of a symmetric molecule): along such a mode the criterion changes by 10^{-6} over a radian, its gradient is at the noise of that flatness and its curvature is nothing, so a Newton step along it is radians long and leads nowhere the criterion can distinguish. The shift bounds that step and takes the mode out of the convergence measure; the modes with a curvature well above it converge at a rate of shift/curvature per step.

The CP-L equations of the DLPNO-MP2 gradient drop the same modes below 3×10^{-4}.
TYPE: Double
DEFAULT: 1.0e-3
OPTIONS: n User-defined value.

- **SAD_CHOLESKY_TOLERANCE**
  SAD guess density decomposition threshold.
  TYPE: Double
  DEFAULT: 1.0e-7
  OPTIONS: n User-defined tolerance value.

- **SAD_E_CONVERGENCE**
  Convergence criterion for the energy in SAD guess.
  TYPE: Double
  DEFAULT: 1.0e-5
  OPTIONS: n User-defined tolerance value.

- **SAD_D_CONVERGENCE**
  Convergence criterion for the density matrix in SAD guess.
  TYPE: Double
  DEFAULT: 1.0e-5
  OPTIONS: n User-defined tolerance value.

- **MAX_DIIS**
  Maximum number of vectors used in the DIIS procedure.
  TYPE: Integer
  DEFAULT: 6
  OPTIONS: n User-defined value.

- **MIN_DIIS**
  Minimum number of vectors used in the DIIS procedure.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: n User-defined value.

- **PRINT_LEVEL**
  Controls the amount of extra information to be printed.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **SCF_MAXITER**
  Maximum number of iterations in the SCF procedure.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **LOCAL_MAXITER**
  Maximum number of iterations in the localization procedure.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **LOCAL_NEWTON_MAXITER**
  Maximum number of second-order steps (rational-function and Newton together) in the polish of the Foster-Boys criterion (`LOCAL_NEWTON`). Three or four are needed from a converged sweep; twenty from a saddle.
  TYPE: Integer
  DEFAULT: 30
  OPTIONS: n User-defined value.

- **SAD_MAXITER**
  Maximum number of iterations in the SAD guess.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **SAD_MAX_DIIS**
  Maximum number of vectors used in the DIIS procedure in SAD guess.
  TYPE: Integer
  DEFAULT: 6
  OPTIONS: n User-defined value.

- **SAD_MIN_DIIS**
  Minimum number of vectors used in the DIIS procedure in SAD guess.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: n User-defined value.

- **SAD_PRINT**
  Controls the amount of extra information to be printed in SAD guess.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **DIIS_ALGORITHM**
  The algorithm to store DIIS intermediates.
  TYPE: String
  DEFAULT: DISK
  OPTIONS: DISK, MEM

  - DISK: Store intermediates in DISK.
  - MEM: Store intermediates in the core memory.

- **GUESS**
  Initial orbital guess for the SCF procedure.
  TYPE: String
  DEFAULT: GWH
  OPTIONS: GWH, SAD, CORE
  - GWH: Generalized Wolfsberg-Helmholtz guess.
  - SAD: Superposition of atomic density.
  - CORE: Core Hamiltonian guess.
- **ORTH_TYPE**
  The orthogonalization algorithm used for the overlap matrix.
  TYPE: String
  DEFAULT: CANONICAL
  OPTIONS: CANONICAL, SYMMETRIC
  - CANONICAL: Canonical orthogonalization.
  - SYMMETRIC: Symmetric orthogonalization.
- **SCF_ALGORITHM**
  Controls the SCF algorithm.
  TYPE: String
  DEFAULT: ROOTHAAN
  OPTIONS: ROOTHAAN, ASO, QC
  - ROOTHAAN: Roothaan algorithm.
  - ASO: Approximately second-order SCF.
  - QC: Quadratically convergent SCF.
- **LOCAL_TYPE**
  Controls the orbital localization algorithm.
  TYPE: String
  DEFAULT: BOYS
  OPTIONS: BOYS, PIPEK_MEZEY
- **ROHF_CANONIC_TYPE**
  Controls the orbital localization algorithm.
  TYPE: String
  DEFAULT: GUEST
  OPTIONS: GUEST, ROOTHAAN, MCWEENY, GAMESS, DAVIDSON, BINKLEY, FAEGRI, PLAKHUTIN, ALPHA, BETA
- **RETRY**
  If SCF iterations are not converged, MacroQC tries different SCF algorithms and initial guesses.
  TYPE: BOOL

DEFAULT: TRUE
OPTIONS: TRUE, FALSE

9 OPT : Geometry Optimization

Code Author: U. Bozkaya, with contributions from B. Ermiş.

9.1 Overview

`MacroQC` performs geometry optimizations (minimization and first-order saddle points, the latter one is experimental at this stage) for a variety of molecular structures using either analytic or numerical energy gradients. `MacroQC` has a rich analytic gradient package; hence, in most cases, one can take advantage of analytic gradients. The present version of `MacroQC` uses Cartesian coordinates in geometry optimizations. `MacroQC` has several optimization algorithms, such as steepest-descent, quasi-Newton, and rational-function optimization algorithms [37] [38] [39].

9.2 Relaxed scans

A potential energy surface scan drives one internal coordinate through a series of values, which is how a torsional profile, a dissociation curve or a reaction coordinate is mapped out. It is set up with `jobtype scan` (or `pes_scan`) and a `$set_scan` block, and it comes in the two kinds the name usually covers.

A *relaxed* scan, SCAN_RELAXED = TRUE, optimizes everything else at each point. It is little more than a loop over the constrained optimizer of Section 9.9.1: the scanned coordinate is a constraint whose value moves from point to point, and each point starts from the structure the previous one ended at, so the path is smooth and every point after the first is cheap.

A *rigid* scan, SCAN_RELAXED = FALSE, computes one energy per point at the input structure distorted along the scanned coordinate and nothing else. The geometry is built rather than optimized: the molecule is moved as two rigid pieces hinged on the coordinate, a distance translating one side along the axis and an angle or a dihedral rotating one side about an axis, which is what a Z-matrix does to its dependent atoms. Every other internal coordinate keeps the value the input geometry gave it. The two kinds answer different questions — a rigid point always lies above the relaxed point at the same value, since the relaxed energy is a minimum over the coordinates the rigid scan holds fixed — and a rigid scan of a coordinate inside a ring is refused, there being no way to move part of the molecule rigidly then.

The energies are collected in a table at the end and the geometries are written to `input_prefix_scan_geoms.xyz`.

9.3 Freezing string method

Finding a first-order saddle point with a surface-walking algorithm requires a starting structure that already lies in the quadratic basin of the transition state, and producing one by hand is the step that fails most often. The freezing string method (FSM) [40] removes the guesswork: it takes the reactant and the product — nothing else — and returns an approximation to the reaction path, whose highest node is a transition-state guess. It is run with `jobtype fsm`.

The product structure is given either in a `$set_product` block, written exactly like the molecule block and accepting everything it accepts, or in an xyz file named by the FSM_PRODUCT_FILE option. Both structures must list the same atoms, and atom a of the reactant is interpolated into atom a of the product, so the order matters: if the two are numbered differently the string is built between structures that do not correspond, and the barrier that comes out can be wrong by orders of magnitude without anything looking amiss. The numbering is therefore checked and, if need be, repaired before anything else happens — see FSM_ATOM_MAP. The product is then rotated and translated onto the reactant at maximum coincidence, since otherwise the straight line between the two runs through a rigid rotation and every interpolated structure along it is chemically meaningless.

The string is then grown a pair of nodes at a time, inwards from both ends. Each new pair is placed one interpolation distance $s = |\mathbf{x}_R - \mathbf{x}_P|/\mathrm{FSM_NNODES}$ in from the innermost pair, relaxed for at most FSM_NGRAD steps in the space perpendicular to the path, and *frozen* for the rest of the run. Freezing is what makes the method cheap: gradients are never spent settling the parts of the string that are far from the barrier and do not describe the transition state anyway. The string stops when its two ends meet, which typically costs of the order of fifty gradients in total.

Three interpolation schemes are available. Straight Cartesian interpolation, FSM_INTERP = CART, is the simplest and the worst: it compresses bonds of groups that merely rotate along the path, and the nodes then start from high-energy structures the few perpendicular steps cannot repair. Linear synchronous transit [41, 42], FSM_INTERP = LST and the default, interpolates all internuclear distances instead and reconstructs the Cartesian structure that carries them by least squares, weighted as r^{-4} so that the short distances — the bonds — are the ones reproduced most closely.

The third, FSM_INTERP = RIC, interpolates the internal coordinates themselves, following Zimmerman [43]. LST holds the distances but knows nothing of angles or torsions, so a path that is mostly a rotation still travels through structures no molecule adopts; bond stretches, bends and torsions are the coordinates chemistry is actually written in, and a straight line in them is a curve in space that keeps its bonding sensible. The coordinate set is the *union* of the primitives of the two endpoints, which is what makes it a reaction rather than a deformation: a bond only the product has is a coordinate from the start, so forming and breaking happen along coordinates of their own instead of by conspiracy among the Cartesians. Every structure along the string is then described in that one set, and each image is reached from the one before it by the usual iterative back-transformation.

Which of the three is best is a property of the reaction, and RIC is not a general improvement

on LST. It is worth being blunt about that, because the argument for interpolating in internal coordinates is a good one and the measured results are mixed.

What was measured here, all with FSM. On the $CH_3CHO \longrightarrow CH_2CHOH$ hydrogen shift, whose endpoints are 3.77 Å apart, RIC's interpolated path is shorter than LST's — 4.56 against 4.81 Å — because it does not have to bend anything back, and with twenty nodes RIC is the only setting of the freezing string that reaches the true transition state at all: every LST spacing tried, up to thirty nodes, converges instead on a spurious saddle of the product, as Section 9.3 describes. That is not a saving of gradients but a difference of answer, and it is the strongest case for RIC in this manual. On $SiH_2 + H_2 \longrightarrow SiH_4$ the two are indistinguishable, 31 gradients each. On $H_2CO \longrightarrow H_2 + CO$ RIC fails and LST takes 47. On the Cope rearrangement of 1-hexen-5-yne, at two different node spacings, LST finds the saddle in 28 gradients and RIC's guess sends the P-RFO search downhill into the product basin instead — and the string that produced it looks perfectly healthy, with a barrier within 3 kcal mol^{-1} of LST's and an approximate Hessian with exactly one negative eigenvalue. One decisive win, one tie, two losses. LST remains the default, on the two losses; RIC is what to reach for when LST's path is visibly unphysical, and what to try before believing an LST saddle whose imaginary mode is not the one the reaction needs.

The reason RIC fails on $H_2CO \longrightarrow H_2+CO$ is understood and is a property of the coordinates rather than of the implementation: the reaction *separates*, and what is left to travel at the end of such a path is the relative orientation of two pieces that no longer share a bond, which internal coordinates describe barely or not at all. The interpolation says so, reporting how far short of the product it stopped. Two other situations make RIC unavailable and fall back to LST by themselves, again with a note in the output: a coordinate set that does not span the internal degrees of freedom at either endpoint, which is what a linear reactant such as HCN gives, and one that has gone singular at the product. Setting PRINT_LEVEL to 1 writes the whole interpolation to `<prefix>_ric_interp.xyz`, which is the thing to look at when a string comes out somewhere unexpected.

Two perpendicular optimizers are available as well. FSM_OPT = CG is the scaled conjugate-gradient scheme of the original paper [40]; FSM_OPT = BFGS, the default, is the quasi-Newton line search of Ref. [44], which reaches a string of the same quality with roughly a third fewer gradients.

Finally, and this is the second half of Ref. [44], FSM builds an *approximate Hessian* at its guess and hands both to a P-RFO saddle search. A model force field in redundant internal coordinates is pushed out to Cartesians with the B matrix, its curvature along the string tangent is stripped out, and the curvature the string itself measured there — a three-point second difference through the guess and its two neighbours — is put in its place. The result has exactly one negative eigenvalue, along the reaction coordinate, by construction; no second derivatives of the energy are computed at any point. This matters because an exact Hessian costs one power of system size more than a gradient, and at a guess geometry it is not even guaranteed to have the right structure. Set FSM_TS_SEARCH = FALSE to stop after the string, and FSM_HESS = FALSE to skip the Hessian construction with it.

The string is written to `input_prefix_fsm_string.xyz` and the approximate Hessian to `input_prefix.hess`, in the format READ_CART_HESS reads, so a saddle search can also

be launched by hand later.

Numbers. Four reactions of the test suite of Ref. [44] are in `tests/fsm`. In each, the whole job — string, guess, Hessian and saddle search — computes no second derivatives at any point, and the saddle it reaches has exactly one imaginary frequency:

reaction	level	gradients	P-RFO cycles	$\nu^{\ddagger}/\mathrm{cm}^{-1}$
$\mathrm{HCN} \rightleftharpoons \mathrm{HNC}$	scf/sto-3g	30	5	$1248.66i$
$\mathrm{SiH_2} + \mathrm{H_2} \rightleftharpoons \mathrm{SiH_4}$	b3lyp/6-31G	31	7	$1325.92i$
$\mathrm{H_2CO} \rightleftharpoons \mathrm{H_2} + \mathrm{CO}$	b3lyp/6-31G	47	19	$1833.80i$
$\mathrm{CH_3CHO} \longrightarrow \mathrm{CH_2CHOH}^{\dagger}$	hf/sto-3g	63	22	$2714.88i$

†with FSM_INTERP = RIC and twenty nodes, and not with the defaults; see below.

For $\mathrm{HCN} \rightleftharpoons \mathrm{HNC}$ the search takes as many cycles from the approximate Hessian as it does from an exact one, which has to be computed first.

The ethanal row is a warning as much as a result. At the default LST interpolation this reaction is where the freezing string is easiest to get a wrong answer out of. The string's highest node lands 130 to 180 kcal mol^{-1} above the reactant, far above the real barrier of 109, and the P-RFO search started there slides off the ridge and converges — cleanly, and with exactly one imaginary frequency — on vinyl alcohol itself, the hydrogen already transferred at C−H = 2.762 Å, whose single imaginary mode at −389 cm^{-1} is the O−H torsion of the enol. That is a first-order saddle and it is not the transition state of this reaction. It happens at every LST spacing tried, up to thirty nodes and 217 gradients. FSM_INTERP = RIC with twenty nodes reaches the real one, in fewer gradients than the LST run that gets it wrong, and so do both growing-string modes (Section 9.6).

The general lesson is worth stating on its own: *a converged saddle search with one imaginary frequency is not evidence that the right saddle was found. Only the mode is.* Look at the imaginary mode, or run an IRC.

Two things about the endpoints are worth knowing, because both are the user's responsibility and neither is a defect of the method. The atoms must correspond one to one between the two structures, and *which* atom is mapped onto which decides how long the path is: mapping the far methyl hydrogen of acetaldehyde onto the hydroxyl hydrogen of vinyl alcohol makes that one atom travel 3.9 Å across the molecule. And a symmetric pair of endpoints asks for a symmetric path: placing the departing $\mathrm{H_2}$ of formaldehyde straight below the carbon relates the two structures by a C_{2v} path, which is a ridge rather than a valley, and the string rides 170 kcal mol^{-1} up it.

9.4 Growing string method

The third double-ended method here, and it is worth saying at the outset where it sits between the other two, because the choice between them is a real one.

The freezing string of Section 9.3 never revisits a node: each is relaxed a few steps and frozen. That is why it is cheap — fifty gradients is a typical whole run — and it is also why what it produces is a *guess* that a saddle search has to finish. The elastic band of Section 9.7 starts from a complete path and relaxes all of it at once, which converges the reaction path but spends gradients on the parts far from the barrier before it knows where the barrier is. The growing string [45, 43], run with `jobtype gsm`, is the middle of the two: it adds nodes from both ends as the string converges, so like FSM it does not populate a path it has not yet found, and unlike FSM it keeps optimizing what it has, so what comes out is a reaction path rather than an interpolation through one.

The endpoints are taken exactly as FSM takes them — a `$set_product` block or an xyz file named by GSM_PRODUCT_FILE — and the product is renumbered and aligned onto the reactant the same way, with the same caveats (Section 9.3).

Growth begins with the two endpoints and one node grown in from each. A side then grows again only when its frontier node has relaxed to GSM_GADDMAX, which is the whole of what makes the string *grow* rather than appear: a node is never placed in a region the string has not yet learned the shape of. Each new node is put one of the remaining shares of the way from the frontier towards the other side, measured along the tangent, and the tangent is a displacement in the union coordinate set of Section 9.3 — the same one FSM_INTERP = RIC uses. Between growth steps every node takes GSM_OSTEPS optimization steps in the space perpendicular to its own tangent, and the string is then reparameterized: each node slides along its tangent until the spacing is even again. That costs no gradients and, because the motion is along the path and not across it, it cannot change the path — it changes where the path is sampled.

How a node moves. The published method takes a scaled steepest-descent step with a conjugate-gradient memory, $\Delta\mathbf{U} = -\mathbf{g}_\perp/\alpha$, and says of it that it "could in principle be accelerated further by using an optimizer that includes Hessian information". What is implemented here is that accelerated optimizer, because the plain one does not merely converge slowly: a step of fixed length along a gradient zig-zags across a narrow valley, and on HCN $\longrightarrow$ HNC the total perpendicular gradient settled into a clean two-cycle and stayed in it for forty iterations and eight hundred gradients, converging in the end only because the P-RFO search rescued it — which is the one thing a growing string exists to make unnecessary.

A node therefore moves by a level-shifted quasi-Newton step in internal coordinates. The gradient is carried into the union coordinate set as $\mathbf{g}_q = \mathbf{G}^{-}\mathbf{B}\mathbf{g}_x$. The space the node is allowed to move in is built by putting the reaction tangent *first* and orthogonalizing the delocalized internal coordinates [46] against it, then dropping it: what is left spans exactly the perpendicular directions, so the constraint is exact and not a penalty — a step built from this basis has no component along the path at all. The Hessian in that basis is $\mathbf{V}^T\mathbf{H}_q\mathbf{V}$, and its eigenvalues are shifted by $\lambda = -e_0 + 0.015$ when the lowest is negative and by 0.005 when it is not, so that $\Delta q_i = -g_i/(e_i + \lambda)$ is downhill in every direction even where the surface is not convex. Each component is capped at GSM_MAXAD and the whole step at a trust radius, which starts at GSM_DMAX and thereafter shrinks when a step makes the gradient

worse and grows when it does not.

$\mathbf{H}_q$ is kept in the *primitive* coordinates rather than in the basis it is used in — a model force field (the OPT module's HESS_GUESS) once and BFGS thereafter, updated with the step the back-transformation actually realized rather than the one that was asked for. This is what makes it survive: the basis is rebuilt at every step, since the tangent turns as the string moves, and a Hessian expressed in a basis that no longer exists is worthless. In primitive coordinates it means the same thing at every step.

Where internal coordinates are unavailable — GSM_INTERP = CART, or a union set that cannot describe both endpoints — the node falls back on the published conjugate-gradient step, with the same trust radius wrapped around it.

Once the two ends meet the tangents become Henkelman's upwind tangents [47], which point at whichever neighbour is uphill and are what keeps a chain of states from folding on itself near the barrier. The string is then relaxed until the sum of the perpendicular gradient magnitudes falls below GSM_TOL_F.

The last piece is the *climbing image* [48]. Once the string has nearly settled its highest node is released along the reaction coordinate and pushed *up* it by $\mathbf{g}_C$, the one component every other node has projected out, while it goes on relaxing across everything else. That ascent is capped like any other component of the step and damped while the Hessian is still a guess. It costs essentially nothing — the gradient it climbs on is the one that node was going to compute anyway — and it is the difference between handing a P-RFO search a structure it converges from and one it does not. The reparameterization leaves the climbing node where it is; everything else keeps being spaced around it.

What this costs, and when it is worth it, deserves plain numbers. GSM asks for several times what FSM does: on the $SiH_2 + H_2 \longrightarrow SiH_4$ test a nine-node string converges in 266 gradients against the 31 the freezing string spends on its guess, and its climbing node puts the barrier at 12.04 kcal mol^{-1} where the raw freezing string maximum was 13.25 and the converged answer is 9.59. Both are then refined by the same P-RFO search to the same saddle, in 7 cycles from either start. What GSM buys is not fewer cycles at the end but a starting structure that came from a path being optimized rather than from an interpolation: FSM is the right first thing to try, and it usually works; GSM is what to reach for when it does not — when the guess is in the wrong place, when the P-RFO search walks off, or when the reaction path itself is wanted and not only the saddle.

Tangents in Cartesian coordinates are available as GSM_INTERP = CART, which is the GSM-CART of Ref. [43]; it costs between two and six times as many gradients there. It is also what happens by itself, with a note in the output, when the union coordinate set cannot describe both endpoints — a linear reactant being the case that does it — because a method that refuses to run is worse than a slower one.

9.5 Single-ended growing string

Everything so far needs the product, and needing the product is a real limitation rather than a formality. For a reaction being *explored* the product is the answer and not the input;

and even when it is known it has to be handed over with its atoms numbered to match the reactant, which Section 9.3 spends several pages repairing and cannot always repair — on the Cope rearrangement the matcher has two mechanisms to choose between and no basis for choosing.

The single-ended growing string [49, 50] asks instead for a few *driving coordinates*, and it is selected by giving them: a `$set_driving` block, one coordinate per line, turns `jobtype gsm` into SE-GSM and no product structure is read at all.

```
$set_driving
  add      2 5              form a bond between atoms 2 and 5
  break    1 2              break the one between 1 and 2
  angle    1 2 3   109.5    drive the valence angle to 109.5 degrees
  torsion  1 2 3 4 180.0    drive the dihedral to 180 degrees
$end_driving
```

Atom numbers are 1-based and targets are in degrees. ADD and BREAK take no target, for the reason given below.

The coordinate set. A double-ended run takes the union of the two endpoints' primitives because the reactant's own set has no coordinate along which the reaction happens. A single-ended run has the same problem and no product to solve it with, so the drivers solve it: every atom pair they name is forced into the bond list *before* the bends and torsions are generated, which puts the stretch in the set and brings the bends and torsions that pair takes part in along with it. A driving angle contributes its two legs, a driving torsion its three.

The driving tangent. There is no second structure to difference against, so the direction the string grows in is built from the drivers alone. Writing d_{ij} for the current distance and R_i for a covalent radius,

$$\text{ADD:}\quad c \;=\; \max\!\big(0,\; d_{ab} - d^{0}_{ab}\big), \qquad d^{0}_{ab} = 0.93\,(R_a + R_b), \tag{7}$$

$$\text{BREAK:}\quad c \;=\; -0.3\max(1, n_{\text{add}}) \;\text{ while } d_{ab} < 5.2\,(R_a + R_b), \quad -0.3 \text{ after,} \tag{8}$$

$$\text{ANGLE, TORSION:}\quad c \;=\; -(\theta_{\text{target}} - \theta), \tag{9}$$

the torsion difference folded onto $(-\pi, \pi]$, and the whole vector normalized. Each of the three is worth a sentence. ADD has a target because a bond that is being formed has a length; the max stops it pulling once the distance is reached instead of pushing back. BREAK has none, because there is no distance at which two atoms are meaningfully apart and a target would only decide how far past the barrier the string kept going; the n_{add} factor is what keeps a concerted reaction concerted, since an unweighted break tangent is outrun by the forming bond and the string walks over an ionic intermediate instead. The sign convention is the opposite of what it looks like: the step taken is $-\Delta q\, c$, so a *positive* component shrinks that coordinate.

The length of the tangent before normalization is how far there still is to drive, and it sets the step length: long while there is a long way to go, short near the end, so that the string does not stride over the barrier it is looking for.

Growth, and where it stops. Only the frontier node is optimized while the string grows, and it is optimized hard — up to GSM_SE_OSTEPS steps, with an exit at GSM_SE_TOL — because the next node is placed by stepping from it and strain left there is carried forward. The nodes behind it are not touched until the whole string is relaxed afterwards. There is no reparameterization: the spacing is set by the driving step and is uneven on purpose.

Growth stops not when the nodes run out but when the string has *walked over a barrier*, which is tested three ways. The cleanest is the sign of $\mathbf{c} \cdot \mathbf{g}$ at the frontier: the step is $-\Delta q\,\mathbf{c}$, so $\mathbf{c} \cdot \mathbf{g} > 0$ says the next driving step goes downhill. Failing that, the energy profile is examined for nodes past its maximum that lie below it by more than 5 $\mathrm{kcal\,mol^{-1}}$ (one suffices) or 3 (two are needed); and failing that, a profile that has come down at all together with a connectivity that is no longer the reactant's. A profile that is monotonically uphill and either very high or flattening out is reported as a dissociation, because that is what it is.

The last node. A string that stops the moment it detects a barrier stops at the top of one, and the last node of a string is an *endpoint* — held fixed while everything between the ends relaxes. A string whose maximum is its own endpoint has nothing for the climbing image to climb. So the end is then relaxed *freely*, an ordinary minimization for a few steps, which drops it into the product basin and puts the maximum back inside the string. It also answers the question the whole method was asked: that structure is the product.

From there the run is identical to a double-ended one — upwind tangents, reparameterization, the climbing image, the approximate Hessian, P-RFO.

What it costs. On HCN $\rightleftharpoons$ HNC at SCF/STO-3G, driving `add 1 3` and `break 1 2` from a slightly bent reactant, the string converges in 111 gradients and the saddle search reaches the exact saddle in 5 cycles. On $SiH_4 \longrightarrow SiH_2 + H_2$ at B3LYP/6-31G, driving `add 4 5`, `break 1 4`, `break 1 5`, it takes 935 gradients against the 266 the double-ended string spends with the product in hand, and arrives at the same saddle to 1×10^{-9} E_h. That ratio is the price of not knowing the answer, and there are reactions for which it is the only price available.

No Cartesian fallback. A driving coordinate *is* an internal coordinate, so there is nothing to fall back on when the coordinate set fails the spanning test of Section 9.3, and the run stops with a message rather than pretending. A fully linear reactant is the case that does it: it carries no bending coordinate at all, so there is nothing for a migrating atom to swing through. Start from a slightly bent structure — five degrees is enough.

9.6 Which of the three

Four reactions, three methods, the same reactant geometry and the same level of theory in every row. "Gradients" is the string plus the P-RFO cycles that follow it, each cycle being one gradient.

Reaction	Method	Gradients	$\nu_{\text{imag}}/\text{cm}^{-1}$	Saddle
HCN $\longrightarrow$ HNC	FSM	36	-1249	correct
SCF/STO-3G	DE-GSM	108	-1249	correct
	SE-GSM	116	-1249	correct
$H_2CO \longrightarrow H_2 + CO$	FSM	38	—	correct
B3LYP/6-31G	DE-GSM	611	—	correct
	SE-GSM	—	—	*symmetry trap*, see below
	SE-GSM*	353	—	correct
$CH_3CHO \longrightarrow CH_2CHOH$	FSM	87	-389	**wrong saddle**
SCF/STO-3G	FSM†	85	-2715	correct
	DE-GSM	1173	-2719	correct
	SE-GSM	1109	-2719	correct
$SiH_2 + H_2 \rightleftharpoons SiH_4$	FSM	37	-1326	correct
B3LYP/6-31G	DE-GSM	301	-1326	correct
	SE-GSM	1107	-1326	correct

*driving one C−H break instead of both; †FSM_INTERP = RIC with twenty nodes, which is what `tests/fsm` uses.

Cost. FSM is three to thirty times cheaper than either growing string, everywhere, and nothing in what follows changes that. It is the right first thing to try.

Reliability, and the case that decides it. What FSM produces is a *guess*, and where the saddle search cannot rescue the guess it returns a wrong answer quietly. $CH_3CHO \longrightarrow CH_2CHOH$ is that case. The freezing string's highest node sits 177 kcal mol^{-1} above the reactant; the P-RFO search started there falls 154 kcal mol^{-1}, slides off the ridge, and converges on a structure with C_1−H = 2.762 Å and O−H = 0.992 Å — vinyl alcohol itself, with the hydrogen already transferred. It is a genuine first-order saddle, with one imaginary frequency, but at -389 cm^{-1} it is the O−H torsion of the enol and not the reaction. The growing string climbs to the hydrogen in flight, C_1−H = 1.475 Å and O−H = 1.163 Å at -2719 cm^{-1}, which is the 1,3-hydrogen shift. Both GSM modes reach it, and so do the double-ended runs with GSM_INTERP = RIC and CART and with nine nodes and eleven — four independent settings, one answer.

The freezing string is not helpless here, and the row marked † is the point: with FSM_INTERP = RIC and twenty nodes it reaches the same saddle in 85 gradients, cheaper than

any of them. What it cannot do is *tell you* which of its two answers is the right one, and that is the difference the growing string is worth paying for — or, more cheaply, the reason to look at the imaginary mode before believing any of them.

So. Try FSM first; it usually works and it is cheap. Reach for DE-GSM when the product is known and FSM's guess is not good enough — when the saddle search walks off, when the imaginary frequency is not the mode the reaction needs, or when the reaction path itself is wanted. Reach for SE-GSM when the product is not known, or when it is known but the atom correspondence of Section 9.3 cannot be settled.

One rule for driving coordinates. Do not drive symmetry-equivalent coordinates symmetrically. $H_2CO \longrightarrow H_2 + CO$ driven as `add 3 4`, `break 1 3`, `break 1 4` from the C_{2v} reactant keeps the string on the symmetry plane for the whole run: it reaches C−H = 1.494/1.494 Å, which is a ridge, while the true saddle is 1.751/1.093. Nothing perpendicular to the plane ever has a gradient, so nothing pulls the string off it. Driving one C−H break instead of both breaks the symmetry and reaches the saddle in 353 gradients.

9.7 Nudged elastic band

Where the freezing string of Section 9.3 spends as few gradients as it can and hands back a guess, the nudged elastic band converges the reaction path itself, and with it the saddle point. It is run with `jobtype neb` and takes its endpoints exactly as FSM does — a `$set_product` block or an xyz file named by NEB_PRODUCT_FILE — and aligns the product onto the reactant the same way.

A chain of NEB_NIMAGES movable images is placed between the two fixed endpoints, by default along the LST interpolation. Every image is then relaxed under a force that is the true force *perpendicular* to the path plus a spring force *along* it,

$$\mathbf{F}_i \;=\; k\big(|\mathbf{R}_{i+1}-\mathbf{R}_i| - |\mathbf{R}_i-\mathbf{R}_{i-1}|\big)\hat{\mathbf{t}}_i \;-\; \Big[\nabla E(\mathbf{R}_i) - \big(\nabla E(\mathbf{R}_i)\cdot\hat{\mathbf{t}}_i\big)\hat{\mathbf{t}}_i\Big]. \tag{10}$$

That projection is the method [51]. Without it the springs pull the band off a curved path — "corner cutting" — and the true force slides the images down into the minima, leaving none where they are wanted; with it, the spring constant can be varied over orders of magnitude without moving the converged band at all.

The tangent is the one-sided estimate of Ref. [47], taken towards the *higher-energy* neighbour rather than as a bisector of the two. This is not a refinement but a repair: with the bisector an image displaced by δx off a straight path feels a destabilizing force $\delta x\, F/2R$ from the slope along the path against a restoring force $\delta x\, C$, so the band is unstable once $F > 2CR$ — and since R is the image spacing, *any* band becomes unstable if enough images are used. Kinks then form and oscillate and the band never converges. The one-sided tangent has no such term. At a maximum or a minimum along the path, where both neighbours lie the same way, the two candidate tangents are mixed by their energy differences so that the tangent turns over smoothly.

By default one image climbs [48]: after NEB_CLIMB_START iterations the highest image is given the full force with its component along the band inverted,

$$\mathbf{F}_{i_{\max}} \;=\; -\nabla E(\mathbf{R}_{i_{\max}}) \;+\; 2\big(\nabla E(\mathbf{R}_{i_{\max}})\cdot\hat{\mathbf{t}}\big)\hat{\mathbf{t}}, \tag{11}$$

and no spring force at all. It walks up the path and down everything perpendicular to it, and converges on the saddle point itself rather than somewhere near it — at no extra cost, since all the images are being relaxed anyway. Climbing is not switched on at the first iteration because the band has not yet resolved the direction the image would climb. NEB_CLIMB = FALSE gives the plain method, whose highest image is only wherever the springs left it.

The band is moved by quick-min, the projected velocity Verlet of Ref. [51]: each image carries a velocity and any part of it not pointing along the current force is discarded. NEB_OPT = SD is plain steepest descent, slower and harder to trip up. Convergence is on the largest and the rms component of the force over the movable images. The path is written to `input_prefix_neb_path.xyz`, and the highest image is then passed to the same approximate Hessian and P-RFO refinement the freezing string uses.

Which to use. On $SiH_2 + H_2 \rightleftharpoons SiH_4$ at `b3lyp/6-31G` the two answer different questions and cost accordingly: the freezing string takes 31 gradients and 7 P-RFO cycles, the band 266 gradients and 1 cycle, its climbing image landing within 1.3×10^{-6} E_h of the same saddle. What the extra gradients buy is the converged path: the barrier from the band is 9.60 kcal mol^{-1}, where the raw string maximum was 13.25. Use the string to find a saddle point, the band when the shape of the path is itself the answer.

The endpoints have to be minima, and the first band has to be sane. NEB is less forgiving here than the freezing string, which asks only that the atoms correspond. An endpoint that is not a minimum has images sliding away from it that the springs must then fight. Worse, the initial band has to be a physically possible path: for HCN $\rightleftharpoons$ HNC both relaxed minima are *linear*, every interpolation between two collinear structures is itself collinear, and the initial band therefore carries the hydrogen from one end of the molecule to the other straight through the two heavy atoms — 880 kcal mol^{-1} above the reactant at the first iteration. The real path swings the hydrogen out of the line. The test in `tests/neb` gives up relaxed endpoints to buy a sensible band; the general remedy is to supply a better initial path.

9.8 IRC Overview

In addition to geometry optimization, `MacroQC` provides intrinsic reaction coordinate (IRC) computations for tracing the minimum-energy path connected to a transition-state structure. The path is followed in mass-weighted Cartesian coordinates, $Q_i = \sqrt{m_i}x_i$, which is what makes it an IRC in the sense of Fukui rather than a steepest-descent path in some other metric. The IRC path may be followed in the forward, reverse, or both directions.

Two integrators are available, selected with IRC_ALGORITHM. HPC is the default, being both the more accurate and the cheaper of the two; SGD is what a run falls back on when there is no Hessian to start from.

SGD is the stabilized Euler method: a steepest-descent step of length ALPHA followed by a line search along the bisector of the two gradient directions, as proposed by Ishida, Morokuma and Komornicki and refined by Schmidt, Gordon and Dupuis. It needs gradients only, and costs two gradients and one energy per point of the path.

HPC is the Hessian based predictor–corrector of Hratchian and Schlegel, the integrator used by Gaussian. The predictor is the local quadratic approximation of Page and McIver, which integrates the steepest-descent equation exactly on the quadratic model; the corrector then integrates the same equation numerically on a distance weighted interpolant surface fitted to the two ends of the predictor step, using a Bulirsch–Stoer scheme with Euler substeps. Because the corrector runs on the fitted surface it needs no electronic structure calculation of its own, so a point of the path costs one gradient — less than SGD — and the resulting path is considerably more accurate. The Hessian is taken analytically at the transition state and updated with Bofill's formula along the path (IRC_HESS_UPDATE), which is the 2005 refinement of the method. HPC therefore requires STARTING_POINT=TS and a Hessian file, and a run that has neither falls back on SGD with a warning rather than stopping.

On HCN $\rightleftharpoons$ HNC at HF/STO-3G, measuring how nearly the mass-weighted gradient is antiparallel to the tangent of the path — which is the definition of the IRC, and is exact only on the path itself — the two integrators behave as follows at a step of 0.02 bohr amu$^{1/2}$: SGD deviates by 1.2° on average and HPC by 0.01°. The deviation of SGD does not shrink as the step is reduced, being a property of the bisector construction rather than of the step length, so HPC is the one to use when the path itself matters, as in variational transition state theory or reaction path Hamiltonian work.

For the STARTING_POINT=TS option, the initial direction is the eigenvector of the mass-weighted Hessian belonging to the negative eigenvalue, obtained after translations and rotations have been projected out. The run stops with an error if the Hessian has no direction of negative curvature, since there is then no reaction path to follow; a warning is issued if the starting geometry turns out not to be a stationary point, or if it is a saddle point of order higher than one.

At present, the IRC module uses the Cartesian Hessian stored in the input_prefix.hess file when READ_CART_HESS=TRUE. For transition-state based IRC computations, this Hessian file is required; it is mass-weighted internally, so the file itself holds the ordinary Cartesian Hessian as written by a frequency job. IRC geometries are written to the input_prefix_irc_geoms.xyz file during the calculation.

9.9 Input Examples

*Geometry optimization at the rhf/cc-pvdz level:

```
$set_molecule H2O
    0 1
```

```
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    method scf
    reference rhf
    jobtype opt
$end_globals

$set_opt
   opt_algorithm rfo
   opt_maxiter 100
   tol_rmsG 3.0E-4
   tol_maxG 4.5E-4
   tol_rmsDisp 1.2E-3
   tol_maxDisp 1.8E-3
$end_opt
```

*IRC computation from a transition-state structure:
For this calculation, the `input_prefix.hess` file must be available. The Cartesian Hessian is read first, mass-weighted and projected, and the IRC path is then followed from the selected transition-state mode. The example is the $\mathrm{Cl} + \mathrm{H_2} \longrightarrow \mathrm{HCl} + \mathrm{H}$ hydrogen abstraction, whose transition state is linear; `direction both` walks down to the reactants on one side and to the products on the other.

```
$set_molecule ClH2
    0 2
    Cl   0.00000   0.00000   0.05931
    H    0.00000   0.00000  -1.43051
    H    0.00000   0.00000  -2.34312
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    aux_basis_corr cc-pvdz-ri
    method mp2
    reference uhf
    jobtype irc
$end_globals

$set_irc
```

```
   read_cart_hess true
   starting_point ts
   direction both
   follow_root 1
   num_step 6
   alpha 0.15
$end_irc
```

*Transition state of HCN $\rightleftharpoons$ HNC from the reactant and the product alone:
The freezing string builds the path between the two structures, takes its highest node as the guess, constructs the approximate Hessian there, and runs the P-RFO saddle search from it. No Hessian is computed at any point.

```
$set_molecule HCN
    0 1
    H   0.000000   0.000000  -1.064000
    C   0.000000   0.000000   0.000000
    N   0.000000   0.000000   1.156000
$end_molecule

$set_product
    0 1
    H   0.000000   0.994000   1.170000
    C   0.000000   0.000000   0.000000
    N   0.000000   0.000000   1.170000
$end_product

$set_globals
    basis sto-3g
    method scf
    reference rhf
    jobtype fsm
$end_globals

$set_fsm
   fsm_nnodes 10
   fsm_ngrad 3
   fsm_interp lst
   fsm_opt bfgs
$end_fsm
```

*Starting geometry optimization by reading the Hessian. :
In order to do this, there must be an `input_prefix.hess` file. (FREQ module generates after every frequency computation).

```
$set_molecule H2O
    0 1
```

```
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    method scf
    reference rhf
    jobtype opt
$end_globals

$set_opt
   read_cart_hess true
   opt_algorithm rfo
   opt_maxiter 50
   tol_rmsG 3.0E-4
   tol_maxG 4.5E-4
   tol_rmsDisp 1.2E-3
   tol_maxDisp 1.8E-3
$end_opt
```

*Transition state from a Hessian computed at the first step (Gaussian's `CalcFC`):
No `.hess` file is needed. The $6N$ displaced gradients are computed once, the saddle search starts from the resulting Hessian, and Bofill updates carry it on. The HCN $\rightleftharpoons$ HNC saddle below converges in five iterations in internals, against 21 from the guess Hessian.

```
$set_molecule
 0 1
 N   0.000000   0.000000   0.000000
 C   1.180000   0.000000   0.000000
 H   0.700000   1.100000   0.000000
$end_molecule

$set_globals
  method scf
  basis sto-3g
  jobtype opt
$end_globals

$set_opt
  opt_type ts
  opt_coord ric
  calc_hess true
$end_opt
```

*Restarting an optimization:
Every optimization rewrites `input_prefix_opt.rst` after each step. A run that stopped – out of iterations, out of wall time, killed – is continued from that file by adding `opt_restart true` to the same input, with a larger `opt_maxiter` if that is what stopped it. The geometry in the input is ignored; the file's geometry, Hessian and trust radius are used, and the iteration count continues.

```
$set_opt
   opt_restart true
   opt_maxiter 200
$end_opt
```

9.9.1 Constrained optimization

Some internal coordinates can be held fixed while the rest of the molecule relaxes. This is what a relaxed scan along a reaction coordinate is made of, and it is also the way to refine a structure whose experimental bond lengths are to be kept, or to hold a fragment at a chosen separation.

Two kinds of constraint are available and they differ only in the value held. A coordinate listed in CONST_BOND, CONST_ANGLE or CONST_DIHEDRAL with no value of its own is *fixed* where the input geometry puts it. Given a value in the matching CONST_BOND_VAL, CONST_ANGLE_VAL or CONST_DIHEDRAL_VAL, it is instead *driven* to that value over the first few steps and held there. Lengths follow GEOM_UNITS, angles and dihedrals are in degrees, and a dihedral target is understood on $(-180, 180]$, so a coordinate at 170° asked for −175° turns the short way.

The coordinates are named by atom numbers counted from one, one constraint per row of the array. A constrained distance need not be a bond: a pair the connectivity does not join is added to the coordinate set for the purpose, which is what makes it possible to pull two fragments apart or to hold a hydrogen bond. An angle or a dihedral, on the other hand, has to exist in the set, and one that does not is reported as an error rather than ignored. Constraints require OPT_COORD = RIC, since a constraint is a statement about an internal coordinate.

Constraints are not restricted to single coordinates held at a value. CONST_LC_* holds a linear combination of them, CONST_RANGE_* confines a coordinate to an interval, imposing nothing while it is inside, and CONST_ATOM freezes whole atoms in space, which internal coordinates cannot express at all.

Mechanically, a constrained coordinate is removed from the space the optimizer works in: the projector $P = GG^{-}$ that removes the redundancies is replaced by

$$P \;=\; P' - P'C\,(CP'C)^{-}\,CP', \qquad P' = GG^{-}, \tag{12}$$

with C the diagonal 0/1 matrix selecting the constrained coordinates, so that neither the gradient nor the Hessian carries a component along them, and the displacement that takes

each coordinate to the value it is to be held at is written into the step directly. The optimization is not converged while any constraint is unmet, whatever the gradient does; and while one is still travelling the energy is expected to rise, so no step is rejected and the trust radius is left where it is until every constraint has arrived.

*Formaldehyde with the C=O bond driven to 1.50 Å and the two hydrogens kept at the H···H distance the input geometry has, everything else relaxed:

```
$set_molecule H2CO
    0 1
    C   0.000000   0.000000   0.000000
    O   0.000000   0.000000   1.350000
    H   0.943000   0.000000  -0.545000
    H  -0.943000   0.000000  -0.545000
$end_molecule

$set_globals
    basis cc-pvdz
    method scf
    jobtype opt
$end_globals

$set_opt
   opt_coord ric
   const_opt true
   const_bond [1,2; 3,4]
   const_bond_val [1.50; 1.886]
$end_opt
```

*A relaxed scan is the same input run repeatedly, one CONST_DIHEDRAL_VAL at a time. Here the HOOH torsion of hydrogen peroxide is held at 60 degrees while every bond and angle relaxes:

```
$set_molecule H2O2
    0 1
    O   0.000000   0.725000   0.030423
    O   0.000000  -0.725000   0.030423
    H  -0.805661   0.893439  -0.482839
    H   0.805661  -0.893439  -0.482839
$end_molecule

$set_globals
    basis cc-pvdz
    method scf
    jobtype opt
$end_globals
```

```
$set_opt
   opt_coord ric
   const_opt true
   const_dihedral [3,1,2,4]
   const_dihedral_val [60.0]
$end_opt
```

*A relaxed scan of the HOOH torsion of hydrogen peroxide, every bond and angle optimized at each point. Leaving SCAN_START out starts the scan at the value the input geometry has.

```
$set_molecule H2O2
    0 1
    O    0.000000    0.725000    0.030423
    O    0.000000   -0.725000    0.030423
    H   -0.805661    0.893439   -0.482839
    H    0.805661   -0.893439   -0.482839
$end_molecule

$set_globals
    basis cc-pvdz
    method scf
    jobtype scan
$end_globals

$set_scan
   scan_type dihedral
   scan_atoms [3,1,2,4]
   scan_start 0.0
   scan_stop 180.0
   scan_step 15.0
$end_scan

$set_opt
   opt_coord ric
$end_opt
```

9.10 OPT Module Options

- **OPT_TYPE**
  Type of the optimization.
  TYPE: Sring
  DEFAULT: MIN
  OPTIONS: MIN, TS

  - MIN : Minimum search
  - TS : Transition-state search

  A transition-state search runs in either coordinate system; internal coordinates are the better choice, as they are for a minimum. It should be started from a computed Hessian – read from a frequency job with **READ_CART_HESS**, or computed on the spot with **CALC_HESS** – since without one the mode that is followed uphill is simply coordinate **FOLLOW_ROOT** of the guess Hessian, which is an assumption about the reaction coordinate rather than knowledge of it.

- **OPT_ALGORITHM**
  The geometry optimization algorithm.
  TYPE: Sring
  DEFAULT: RFO
  OPTIONS: RFO, QN, SD

  - RFO : Rational Function Optimization
  - QN : Quasi-Newton
  - SD : Steepest Descent

  RFO is the default because it is never worse than QN by much and is sometimes much better. Over the 23 minimizations of `tests/opt` that can be run either way, the two land on the same structure everywhere – the largest energy difference between them is 1.4×10^{-7} E_h – and RFO takes 128 iterations in total against 185: it wins 6 cases, loses 1, and ties 16. The one large difference is `butane-scf-ric-btretry-opt`, 9 iterations against 62, and it says what the difference is between them. A quasi-Newton step is the full Newton step, which is only as good as the Hessian it is built on and points wherever a poor one sends it; the RFO step solves an eigenvalue problem whose shift keeps the step in a region where the quadratic model is still trusted. That matters where the guess Hessian is worst, which is exactly where an optimization is in trouble.

- **HESS_UPDATE**
  The hessian update algorithm.
  TYPE: Sring
  DEFAULT: BFGS
  OPTIONS: BFGS, BOFILL

  - BFGS : Broyden-Fletcher-Goldfarb-Shannoa Hessian update algorithm.

– BOFILL : BOFILL Hessian update algorithm.

- **PRINT_LEVEL**
  The amount of information to print to the output file.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **OPT_MAXITER**
  The maximum number of geometry optimization steps. A run that reaches this limit without converging is an error: the output reports which convergence tests are still open and prints the last geometry reached, and MacroQC exits with a non-zero return code. The run is continued with **OPT_RESTART** and a larger limit; see there.
  TYPE: Integer
  DEFAULT: 100
  OPTIONS: n User-defined value.

- **FOLLOW_ROOT**
  Root for Partial RFO to follow.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **ALPHA**
  The line search parameter.
  TYPE: Double
  DEFAULT: 1.0
  OPTIONS: n User-defined value.

- **MAX_STEP**
  The trust radius: no single coordinate of the optimization step may exceed it. In Cartesian coordinates the unit is bohr; in redundant internal coordinates each component is capped in its own unit, bohr for a stretch and radians for a bend, an out-of-plane angle or a torsion. The radius is adjusted at every step from the ratio of the actual energy change to the one the quadratic model predicted, within the bounds **MAX_STEP_MIN** and **MAX_STEP_MAX**.
  TYPE: Double
  DEFAULT: 0.5
  OPTIONS: n User-defined value.

- **MAX_STEP_MIN**
  Lower bound of the trust radius. A step whose predicted and actual energy changes disagree badly ($\rho < 0.25$) shrinks the radius by a factor of four, but never below this value.
  TYPE: Double
  DEFAULT: 1.0e-3
  OPTIONS: n User-defined value.

- **MAX_STEP_MAX**
  Upper bound of the trust radius. A step that the quadratic model predicted well ($\rho > 0.75$) doubles the radius, but never beyond this value.
  TYPE: Double
  DEFAULT: 1.0
  OPTIONS: n User-defined value.

- **CONSECUTIVE_BACKSTEPS**
  How many times in a row a step may be rejected. When a step raises the energy of a minimization, the geometry is returned to the last accepted point and the step is retaken at half its length; no gradient is recomputed for the undo. Set to 0 to disable rejection and let the trust radius alone handle bad steps.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: n User-defined value.

- **LEVEL_SHIFT_PARAM**
  Level shift to aid convergence.
  TYPE: Double
  DEFAULT: 0.01
  OPTIONS: n User-defined value.

- **CUTOFF**
  The cutoff value for numerical procedures.
  TYPE: Double
  DEFAULT: 1.0e-10
  OPTIONS: n User-defined value.

- **TOL_MAXDISP**
  Convergence criterion for geometry optimization: maximum displacement (cartesian coordinates, atomic units).
  TYPE: Double
  DEFAULT: 1.8e-3
  OPTIONS: n User-defined value.

- **TOL_RMSDISP**
  Convergence criterion for geometry optimization: rms displacement (cartesian coordinates, atomic units).
  TYPE: Double
  DEFAULT: 1.2e-3
  OPTIONS: n User-defined value.

- **TOL_MAXG**
  Convergence criterion for geometry optimization: maximum gradient (cartesian coordinates, atomic units).
  TYPE: Double

DEFAULT: 4.5e-4
OPTIONS: n User-defined value.

- **TOL_RMSG**
  Convergence criterion for geometry optmization: rms gradient (cartesian coordinates, atomic units).
  TYPE: Double
  DEFAULT: 3.0e-4
  OPTIONS: n User-defined value.

- **DIATOMIC_OPTIMIZER**
  `MacroQC` includes a separate optimizer for diatomic molecules. This optimizer is automatically selected for diatomic molecules. The diatomic optimizer generates 5 displaced geometries around the initial geometry and computes energy at each perturbed geometry. Then, using a 5-point interpolation formula, an analytic potential energy function is generated. Using this analytic potential energy function, geometry optimization, harmonic and anharmonic vibrational frequency computations are performed.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **WRITE_CART_HESS**
  Do write Cartesian Hessian?
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **READ_CART_HESS**
  Read the starting Hessian from `input_prefix.hess`, the Cartesian Hessian a frequency job writes. Not to be combined with **CALC_HESS**.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **CALC_HESS**
  Compute the Cartesian Hessian at the first geometry, by central differences of the gradient over $6N$ displaced structures (the frequency module's machinery, with **STEP_BOHR** as the displacement), and start the optimization from it; the quasi-Newton update carries it on from there. This is Gaussian's `CalcFC`. It costs $6N$ gradients, which a saddle search or a floppy molecule usually earns back: the model force field knows nothing about the coupling between coordinates, and the first steps of a BFGS run are spent learning it. The Hessian is used in either coordinate system, and its translations and rotations are projected out.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **RECALC_HESS**
  Recompute the Hessian every this many iterations (Gaussian's `RecalcFC=N`); the update is skipped on those iterations. Zero means never. With **CALC_HESS** the Hessian is computed at iterations $1, 1+N, 1+2N, \ldots$; without it, at $1+N, 1+2N, \ldots$
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **WRITE_RESTART**
  Rewrite `input_prefix_opt.rst` after every step. The file holds the geometry reached, the Hessian in Cartesian coordinates, the trust radius, the iteration count and, for a saddle search, the mode being climbed (as a Cartesian displacement, so that the restart recognizes it by overlap rather than starting over from **FOLLOW_ROOT**); the gradient is not stored and is recomputed on restart. Since the Hessian is stored in Cartesians the file does not depend on the coordinate set: a restart in redundant internals rebuilds its set at the restart geometry.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **OPT_RESTART**
  Continue an optimization from `input_prefix_opt.rst`. The geometry in the input is ignored in favour of the one in the file, the Hessian and the trust radius are taken from it, and the iteration count continues where it stopped – so a run that hit **OPT_MAXITER** needs a larger limit. The Hessian in the file takes precedence over **READ_CART_HESS** and over **CALC_HESS**, which are then ignored with a note; **RECALC_HESS** keeps counting from the file's iteration. It is an error to ask for a restart without a readable file.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **PROJECT_OUT_RC**
  Do project out redundant coordinate components from Cartesian Hessian?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **OPT_COORD**
  The coordinate system that will be used in optimization.
  TYPE: Sring
  DEFAULT: RIC
  OPTIONS: RIC, CART

  - RIC : Redundant internal coordinates.
  - CART : Cartesian coordinates.

A molecule whose atoms all lie on one line, such as CO_2 or acetylene, is a special case of RIC. A bend of a linear frame has no plane to be measured in, and it needs none: the energy is stationary along every bending direction of a linear molecule, so a gradient-driven step cannot leave the axis in any case. The coordinate set is then the $N-1$ stretches alone, and the optimization keeps the molecule linear. If a structure is meant to bend, start it from a bent geometry or use `OPT_COORD = CART`.

- **TOL_RMSDX**
  Convergence criterion for redundant internal coordinates to cartesian coordinates transformation iterations: $rms(\Delta X)$
  TYPE: Double
  DEFAULT: 1.0e-5
  OPTIONS: n User-defined value.

- **TOL_MAXDX**
  Convergence criterion for redundant internal coordinates to cartesian coordinates transformation iterations: $max(\Delta X)$
  TYPE: Double
  DEFAULT: 1.0e-4
  OPTIONS: n User-defined value.

- **RIC2CART_MAXITER**
  The maximum number of redundant internal coordiantes to cartesian coordinates transformation steps.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **RIC2CART_RETRIES**
  How many times the internal-to-Cartesian back-transformation may halve the internal step and start again before giving up on it. A back-transformation that does not converge nearly always means the requested Δq is too large for the linear model behind each micro-iteration, so asking for less is the first thing to try; it keeps the second-order information that the first-order fallback $\mathbf{x}_1 = \mathbf{x}_0 + A_0\mathbf{s}_0$ throws away, at the cost of one more pass of a loop whose iterations are B-matrix builds rather than energies. A step that had to be halved, or one that failed outright, also shrinks the trust radius. Set to zero to fall back immediately.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: n User-defined value.

- **MAX_STEP_X**
  Maximum Cartesian displacement, in bohr, of one micro-iteration of the redundant-internal to Cartesian back-transformation.
  TYPE: Double
  DEFAULT: 0.1
  OPTIONS: n User-defined value.

- **PRINT_RIC2CART**
  Print the micro-iterations of the internal-to-Cartesian back-transformation. This is an inner loop of every optimization step and it says nothing once the back-transformation works, so it is off by default; the iteration count is still reported on one line per step, and a back-transformation that fails always reports itself in full, whatever this option says. Turn it on to follow a step that is not being reproduced in Cartesian coordinates.
  TYPE: Bool
  DEFAULT: FALSE
  OPTIONS: TRUE/FALSE.

- **HESS_GUESS**
  The starting Hessian for an optimization in redundant internal coordinates, when none is read from file. A diagonal model force field is used rather than the unit matrix, because the two are not comparable: a stretch force constant is around 0.45 a.u. and a torsion around 0.0025, so a unit matrix makes every stretch step several times too short and every torsional step hundreds of times too long. In Cartesian coordinates this option has no effect and the unit matrix is used.
  TYPE: String
  DEFAULT: SCHLEGEL
  OPTIONS: SCHLEGEL, LINDH, FISCHER, UNIT

  - SCHLEGEL : Schlegel's rules, *Theor. Chim. Acta* **66**, 333 (1984).
  - LINDH : the harmonic model function of Lindh *et al.*, *Chem. Phys. Lett.* **241**, 423 (1995), with the parameters Molpro and Molcas use. Needs no covalent radii and makes no bonded/non-bonded decision.
  - FISCHER : Fischer and Almlöf, *J. Phys. Chem.* **96**, 9768 (1992).
  - UNIT : the unit matrix.

- **OOFP_COORD**
  Do generate an out-of-plane coordinate at every three-coordinate centre? A centre with three ligands has six local degrees of freedom and exactly six primitives to describe them, three stretches and three angles, but at planarity the three angles sum to 2π and the set loses rank: nothing in it moves the centre out of the ligand plane. Formaldehyde and BH_3 also generate no torsion at all, so without this coordinate their wagging mode is missing entirely and a pyramidal starting structure cannot be relaxed.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **BPRIME_TERM**
  Do include the $\mathbf{B}'$ term when a Cartesian Hessian read from file is transformed into internal coordinates, $H_q = G^{-}B\,(H_x - \mathbf{B}'^{\mathsf{T}}\mathbf{g}_q)\,B^{\mathsf{T}}G^{-}$? The term vanishes at a stationary point and is evaluated by finite differences of $B^{\mathsf{T}}\mathbf{g}_q$, which costs $6N$ rebuilds of the B matrix once per optimization. It is typically one to five per cent of the Hessian, the larger figure at a transition-state guess where the gradient is large.

TYPE: Boolean
DEFAULT: TRUE
OPTIONS: TRUE, FALSE

- **RIC_REBUILD**
  May the coordinate set be rebuilt during an optimization? The set is generated once, at the input geometry, which is what keeps it continuous: a coordinate that changed meaning half way through would make the Hessian, the step and the convergence test refer to different things from one iteration to the next. It can nevertheless stop matching the molecule. An ordinary valence angle is a poor description of a unit that has straightened — there is no plane left to measure it in — and the two-component linear bend that replaces it is better conditioned. When a bend passes 175° the set is therefore regenerated at the current geometry, with the Hessian carried across through Cartesian coordinates. Propyne started with its C≡C–H end bent 20° converges in 8 iterations with the rebuild and 11 without, to the same energy. Nothing is done where a rebuild could not help: a three-atom near-linear molecule has no atom off the axis to measure a linear bend against, and its bend stays as it is — which is correct, not a compromise, the bend rows being well behaved right up to linearity.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **CONST_OPT**
  Do hold some internal coordinates at fixed values while the rest of the molecule relaxes? A constrained coordinate is removed from the space the optimizer works in — the projector $P = GG^{-}$ is replaced by $P' - P'C(CP'C)^{-}CP'$, with C selecting the constrained coordinates — so the gradient and the Hessian carry no component along it, and the displacement that takes it to the value it is to be held at is written into the step directly. The coordinates named by CONST_BOND, CONST_ANGLE and CONST_DIHEDRAL are constrained; a coordinate given no value of its own is held where the input geometry puts it. Requires OPT_COORD = RIC, since a constraint is a statement about an internal coordinate.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **CONST_BOND, CONST_ANGLE, CONST_DIHEDRAL**
  The coordinates to constrain, as atom numbers counted from one, one constraint per row: `const_bond [1,2; 3,4]` constrains two distances, `const_angle [3,1,2]` one valence angle, `const_dihedral [1,2,3,4]` one torsion. A distance need not be a bond: a pair the connectivity does not join is added to the coordinate set for the purpose, which is what makes it possible to pull two fragments apart or to hold a hydrogen bond. An angle or a dihedral, on the other hand, has to exist in the set, and one that does not is reported as an error rather than ignored.
  TYPE: Integer array
  DEFAULT: none

- **CONST_BOND_VAL, CONST_ANGLE_VAL, CONST_DIHEDRAL_VAL**
  The value each constrained coordinate is to be held at, one per constraint and in the same order. Lengths follow GEOM_UNITS, angles and dihedrals are in degrees, which is the convention of the Z-matrix input. Left out, the coordinate is held wherever the input geometry has it. A coordinate that starts away from its value is driven there over the first few steps rather than in one jump, and no step is rejected while it is on its way.
  TYPE: Double array
  DEFAULT: none

- **CONST_ATOM**
  Atoms held where the input geometry puts them, e.g. `const_atom [1, 2, 7]`. This is the cluster-model case: a piece cut out of a crystal or a protein, with its boundary kept at the positions the experiment gave it and only the reactive part free; and it is how a structure with newly added hydrogens is cleaned up, the heavy atoms frozen and the hydrogens relaxed.

  It cannot be had from constraints on internal coordinates, which carry no absolute position: holding a fragment in the frame its crystal structure gave it would take every internal coordinate among its atoms, and even then the frozen part could still translate and rotate as a whole. Three Cartesian primitives are therefore declared into the coordinate set for each frozen atom — their rows of B are unit vectors — and the ordinary constraint machinery holds them. They are generated only for the atoms being frozen: a set carrying every Cartesian would span the translations and rotations as well, which is what the redundancy projector exists to remove.
  TYPE: Integer array
  DEFAULT: none

- **CONST_RANGE_BOND, CONST_RANGE_ANGLE, CONST_RANGE_DIHEDRAL, CONST_RANGE_LO, CONST_RANGE_HI**
  A coordinate confined to an interval rather than held at a value, $q_{\min} \le q \le q_{\max}$: the atoms are named as for the other constraints and the two arrays carry one bound each per row. Lengths follow GEOM_UNITS, angles and dihedrals are in degrees.

  An inequality is an active-set problem rather than an equality with extra steps. While the coordinate is inside its interval nothing is imposed and the optimization runs free; when it reaches a bound the constraint takes hold there; and it is released again the moment the energy would fall by moving back inside. That last part is the half that is easy to leave out, and a coordinate that is never released stays glued to the bound it touched. The decision is remade at every geometry and the projector is rebuilt when it changes.
  TYPE: Integer array, Double array
  DEFAULT: none

```
$set_opt
   const_opt true
   const_range_bond [1,3]
   const_range_lo [1.05]
   const_range_hi [1.30]
$end_opt
```

- **CONST_LC_BOND, CONST_LC_ANGLE, CONST_LC_DIHEDRAL, CONST_LC_COEFF, CONST_LC_VAL**
  A constraint need not be a single coordinate. These hold a linear combination,

$$\sum_k c_k q_k \;=\; \text{value}, \tag{13}$$

  which is how two bonds are kept equal without imposing symmetry ($r_1 - r_2 = 0$), how a bridge is held at a fixed total length ($r_1 + r_2$), and how a proton transfer is driven along the antisymmetric stretch r(A–H) − r(H–B) that it wants rather than along either bond alone. The rows of the three atom arrays carry a group number first and then the atoms; rows sharing a group number are terms of one constraint, and their coefficients are the entries of CONST_LC_COEFF in the order the rows are read — bonds, then angles, then dihedrals. CONST_LC_VAL carries one value per group and may be left out, in which case the combination is held where the input geometry has it. The terms of one combination have to be of one type, so that its value has a unit at all.
  TYPE: Integer array, Double array
  DEFAULT: none

```
$set_opt
   const_opt true
   const_lc_bond  [1,1,3; 1,1,4]   ! group, then the atoms
   const_lc_coeff [1.0; -1.0]      ! r(1,3) - r(1,4)
   const_lc_val   [0.2]            ! held at 0.2 ang
$end_opt
```

- **CONST_TOL**
  A constraint counts as met when its deviation falls below this, in bohr or radian. The optimization is not converged while any constraint is unmet, whatever the gradient does.
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined value.

- **CONST_WEIGHT, CONST_MAX_STEP**
  How fast a constraint is driven to its value. Per step the correction is bounded by CONST_WEIGHT × max($|\Delta q|_{\text{free}}$, MAX_STEP/2), i.e. relative to the trust radius

and to the step the unconstrained coordinates are taking, so that an imposed motion never dwarfs the motion the gradient asks for; CONST_MAX_STEP is the ceiling that bound may not cross. Raising CONST_WEIGHT reaches the value sooner at the cost of a worse quadratic model on the way.
TYPE: Double
DEFAULT: 1.0 and 0.3
OPTIONS: n User-defined value.

9.11 SCAN Module Options

- **SCAN_TYPE**
  Which kind of coordinate is scanned.
  TYPE: String
  DEFAULT: BOND
  OPTIONS: BOND, ANGLE, DIHEDRAL

- **SCAN_ATOMS**
  The atoms defining it, counted from one: two for a distance, three for an angle, four for a dihedral, e.g. `scan_atoms [3,1,2,4]`. A scanned distance need not be a bond — a pair the connectivity does not join is added to the coordinate set for the purpose, which is what makes a dissociation curve possible.
  TYPE: Integer array
  DEFAULT: none

- **SCAN_RELAXED**
  Is everything else optimized at each point? TRUE runs a constrained optimization per point and costs a full optimization each; FALSE computes a single energy per point at the input structure distorted along the scanned coordinate. Hydrogen peroxide scanned over its HOOH torsion has its minimum at the gauche 120° when relaxed and runs downhill to trans when rigid, which is the difference between the two questions rather than an error in either.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **SCAN_START, SCAN_STOP, SCAN_STEP**
  The range and the spacing. Lengths follow GEOM_UNITS, angles and dihedrals are in degrees. SCAN_STOP has to be given; leaving SCAN_START out starts the scan at the value the input geometry has, which costs nothing since the first point then needs no travelling. SCAN_STEP is a size and its sign is taken from the two ends, so a scan may run in either direction.
  TYPE: Double
  DEFAULT: SCAN_STEP 0.1
  OPTIONS: n User-defined value.

9.12 FSM Module Options

- **FSM_PRODUCT_FILE**
  An xyz file holding the product structure, in angstrom, with the atoms in the same order as the reactant's. It must be enclosed in double quotes. Left out, the product is read from a `$set_product` block of the input instead; one of the two has to be there.
  TYPE: Line
  DEFAULT: none

- **FSM_ATOM_MAP**
  Renumber the product atoms onto the reactant's when the order given is not the best one available. A reaction breaks and makes a few bonds and leaves the rest of the connectivity alone, so the correspondence to look for is the one that conserves as many bonds as possible,

  $$\max_{m} \left|\{(i,j) \in B_R : (m(i), m(j)) \in B_P\}\right|,$$

  over permutations m that map each atom onto one of the same element. It is found from three starting guesses — the order as written, nearest partner after rigid alignment, and matching graph environments in the sense of Weisfeiler and Lehman — each refined by transpositions. The order given is replaced only if the one found conserves *strictly more* bonds, so an input that is already right is never touched, and the mapping applied is printed. Where the bond count cannot choose — the three hydrogens of a methyl group are the standard case — the order written wins: a tied swap is taken only if it leaves more atoms where the input put them, and the shorter travel decides only after that. This matters because a choice like "this hydrogen is the one that migrates" lives in the ordering and nowhere else.

  The criterion has a limit worth knowing, because it is a real one and it is quiet. Conserving the most bonds identifies the correspondence that changes the least, which is what repairs a numbering; it is not a way of choosing a mechanism. Two mechanisms that each change one bond score the same, and the tie then falls to the geometric test, which knows nothing about chemistry. The thermal rearrangement of 1-hexen-5-yne to 1,2,5-hexatriene is the case: the Cope rearrangement that the reaction actually follows breaks a C–C bond and forms another, while a formal 1,3 shift of a hydrogen reaches the same product by moving a C–H. Both conserve 12 of the 13 bonds of the reactant, and the shift is the shorter journey (RMSD 1.50 against 2.15 Å), so it is the one that gets picked — and the string is then built between structures that correspond atom for atom but along a path the reaction does not take. When the mechanism is known, write the product in the order it implies: an ordering that already conserves as many bonds as anything else is kept exactly as given. One renumbering is refused whatever it conserves: the one that makes the product a rigid copy of the reactant. That is the symmetric atom transfer $H + H_2 \longrightarrow H_2 + H$, where exchanging the two outer atoms conserves the one bond there is and leaves no reaction to find a path for; which atom is transferred is not something the connectivity can decide, and the numbering given stands.
  TYPE: Boolean

DEFAULT: TRUE
OPTIONS: TRUE, FALSE

- **FSM_NNODES**
  The interpolation distance is $|\mathbf{x}_R - \mathbf{x}_P|$ divided by this, so it is the number of nodes the finished string would carry if no node ever moved off the interpolation. Larger means a denser string, a better path and more gradients. Ref. [40] recommends 10 and 20 as its large and small spacings, Ref. [44] uses 18 throughout.
  TYPE: Integer
  DEFAULT: 18
  OPTIONS: n User-defined value.

- **FSM_NGRAD**
  How many perpendicular optimization steps — that is, gradients — each new node is allowed. A node may stop earlier, when its energy has settled, when the line search condition is met, or when it has drifted too far from the node growing towards it. Three is the value both papers use; one turns the relaxation off entirely and leaves the pure interpolation.

  Raising it does not always buy anything, and it is worth knowing why before reaching for it. The drift test is the one that usually stops a node: a relaxed node may not end up more than half a node spacing further from its partner than the interpolation left it, and the spacing is $|\mathbf{x}_R - \mathbf{x}_P|$/FSM_NNODES. The two options are therefore coupled — a denser string gives each of its nodes *less* room to relax, not more. Measured on a Cope rearrangement whose nodes were stopping at $|\mathbf{g}_\perp| \approx 0.05$: going from three steps to fifteen left the total gradient count unchanged at fifty, because the drift limit fired first on six nodes and several others stopped after a single step. The column headed `grads` in the string table is the thing to read — if it is full of ones and twos, the allowance is not what is binding.
  TYPE: Integer
  DEFAULT: 3
  OPTIONS: n User-defined value.

- **FSM_INTERP**
  How new nodes are interpolated.
  TYPE: String
  DEFAULT: LST
  OPTIONS: LST, CART, RIC

  - LST : linear synchronous transit, which interpolates internuclear distances. The safe default; see Section 9.3.
  - CART : a straight line in Cartesian coordinates.
  - RIC : a straight line in the internal coordinates of the union of the two endpoints' primitives. Not a general improvement on LST — one win, one tie and two losses over the test set — and the wrong choice on reactions that separate into fragments; see Section 9.3. Falls back to LST when the coordinate set cannot describe both ends.

- **FSM_RIC_TOL, FSM_RIC_MAXITER**
  One image of an RIC interpolation: the Cartesian displacement in bohr that the back-transformation has to get under, and the number of micro-iterations it may spend getting there. Both are deliberately loose, and the defaults are Zimmerman's [43]. The point a back-transformation in redundant coordinates is asked for does not exist exactly — a set of primitives has more members than the molecule has degrees of freedom, so a point on the straight line between two structures satisfies the relations between them only approximately — and the first few iterations, which travel along the tangent where the target is nearly right, are the ones worth having. Converging past that buys a compromise structure that trades a stiff coordinate away for a soft one.
  TYPE: Double, Integer
  DEFAULT: 7.0E-3, 8
  OPTIONS: n User-defined value.

- **FSM_OPT**
  The perpendicular optimizer.
  TYPE: String
  DEFAULT: BFGS
  OPTIONS: BFGS, CG

  - BFGS : quasi-Newton line search with a BFGS inverse-Hessian update, carried from node to node along each side of the string.
  - CG : scaled conjugate gradient, the original FSM.

- **FSM_SCALING**
  The step scaling factor γ of the conjugate-gradient optimizer, in hartree Å^{-2}; the first step is $-\mathbf{g}^{\perp}/\gamma$. Ignored when FSM_OPT is BFGS.
  TYPE: Double
  DEFAULT: 1.75
  OPTIONS: n User-defined value.

- **FSM_MAX_DISP**
  No Cartesian coordinate may move further than this in one perpendicular step, in angstrom.
  TYPE: Double
  DEFAULT: 0.05
  OPTIONS: n User-defined value.

- **FSM_SIGMA**
  The line search stability factor $\sigma \in (0, 1)$ of the BFGS optimizer. A smaller value tightens the line search and follows the reaction path more closely, at the cost of gradients.
  TYPE: Double
  DEFAULT: 0.7
  OPTIONS: n User-defined value.

- **FSM_EPS**
  The energy change per line search step the trial step length aims at, in hartree. When the reactant and the product are closer in energy than ten times this, that difference is used instead.
  TYPE: Double
  DEFAULT: 0.0004
  OPTIONS: n User-defined value.

- **FSM_TOL_E**
  A node stops early once its energy moves by less than this between steps, in hartree.
  TYPE: Double
  DEFAULT: 5.0E-5
  OPTIONS: n User-defined value.

- **FSM_NUM_INTERP**
  How many images the dense LST string carries. The new nodes and their tangents are picked out of it by arclength, so this sets how finely the interpolation distance can be matched.
  TYPE: Integer
  DEFAULT: 100
  OPTIONS: n User-defined value.

- **FSM_LST_TOL, FSM_LST_MAXITER**
  Convergence threshold on $|\partial S/\partial \mathbf{x}|$ and iteration ceiling of the Newton minimization of one LST image.
  TYPE: Double, Integer
  DEFAULT: 1.0E-3, 500
  OPTIONS: n User-defined value.

- **FSM_LST_WEIGHT**
  The weight of the Cartesian anchor term of the LST functional, in atomic units. The distance term of that functional is blind to rigid motion and to reflection, so this term is what makes the least-squares fit determinate at all — and it has to be strong enough to do it. Halgren and Lipscomb quote 10^{-6} [41], which is smaller than every r^{-4} distance weight in the functional and leaves neighbouring images of the dense string free to end up in unrelated frames; since those images are subtracted from each other to obtain arclengths and tangents, the string then misbehaves. On HCN $\rightleftharpoons$ HNC a single image flips and puts a 0.87 Å jump into a 1.36 Å path. The default is Q-Chem's value, at which the same string is smooth to 0.02 Å per image.
  TYPE: Double
  DEFAULT: 1.0E-2
  OPTIONS: n User-defined value.

- **FSM_HESS**
  Build the approximate Hessian at the highest node and write it to `input_prefix.hess`.
  TYPE: Boolean

DEFAULT: TRUE
OPTIONS: TRUE, FALSE

- **FSM_TS_SEARCH**
  Follow the string with a P-RFO transition-state optimization started from its highest node. The OPT module options govern that search as usual; only OPT_TYPE and READ_CART_HESS are set by FSM itself.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

9.13 GSM Module Options

- **GSM_PRODUCT_FILE, GSM_ATOM_MAP**
  The product structure and the repair of its atom ordering, exactly as the corresponding FSM options; see Section 9.3.
  TYPE: Line, Boolean
  DEFAULT: none, TRUE

- **GSM_NNODES**
  Nodes of the finished string, the two endpoints included. Eleven is the value used throughout Ref. [43]. Unlike FSM_NNODES this is a count and not a spacing: the string stops growing when it has this many.
  TYPE: Integer
  DEFAULT: 11
  OPTIONS: n User-defined value.

- **GSM_GADDMAX**
  A new node is added to a side when its frontier node has relaxed to below this perpendicular gradient, in hartree Å^{-1}. Loosening it grows the string faster and on poorer information; tightening it spends more gradients before each addition. If the string stops growing before it is complete, this is usually the option to loosen.
  TYPE: Double
  DEFAULT: 0.1
  OPTIONS: n User-defined value.

- **GSM_TOL_F**
  Convergence of the string: the sum over its nodes of the perpendicular gradient magnitudes, in hartree Å^{-1}. Reference [43] reports its costs at 0.4 and at 0.1, and 0.1 with a climbing image is what reaches the exact saddle in every one of its test cases.
  TYPE: Double
  DEFAULT: 0.1
  OPTIONS: n User-defined value.

- **GSM_OSTEPS**
  Quasi-Newton steps each node takes between reparameterizations. The paper uses

two while the string is growing and three afterwards; one number serves for both here, because the difference is not what decides the cost.
TYPE: Integer
DEFAULT: 3
OPTIONS: n User-defined value.

- **GSM_MAXITER**
  Iterations allowed for the growth phase, and again for the optimization that follows it.
  TYPE: Integer
  DEFAULT: 60
  OPTIONS: n User-defined value.

- **GSM_MAXAD, GSM_DMAX**
  The two ceilings on a quasi-Newton step, as displacements in the internal coordinate set: the largest change allowed in any one coordinate, and the trust radius on the step as a whole. GSM_DMAX is only what a node *starts* with — from there it shrinks when a step makes the gradient worse and grows when it does not, between a fifth and five times its initial value. Both defaults are those of the reference implementation.
  TYPE: Double, Double
  DEFAULT: 0.075, 0.1
  OPTIONS: n User-defined value.

- **GSM_SCALING, GSM_MAX_DISP**
  The step scaling α of the conjugate-gradient fallback, in hartree Å^{-2}, and the ceiling on one Cartesian component of one of its steps in Å. These apply only where the quasi-Newton optimizer cannot run — GSM_INTERP = CART, or a union coordinate set that cannot describe both endpoints — and have no effect on an ordinary run. Reference [43] uses $\alpha = 4.5$ in Cartesians. Smaller scaling optimizes faster and, too small, unstably; the fallback carries a trust radius of its own, between a tenth and two and a half times GSM_MAX_DISP, which is what keeps "too small" from being fatal.
  TYPE: Double, Double
  DEFAULT: 4.5, 0.1
  OPTIONS: n User-defined value.

- **GSM_CLIMB**
  Release the highest node along the reaction coordinate once the string has nearly settled, and let it climb to the saddle; see Section 9.4. There is very little reason to switch it off.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **GSM_INTERP**
  Whether the tangents and the placement of new nodes are internal coordinates or Cartesian. RIC falls back to CART by itself, with a note, when the union coordinate set cannot describe both endpoints.

TYPE: String
DEFAULT: RIC
OPTIONS: RIC, CART

- **GSM_REPARAM_STEPS**
  Passes of the spacing correction after each optimization cycle. One is normally enough; more of them even the string out further before the next cycle sees it.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **GSM_RIC_TOL, GSM_RIC_MAXITER, GSM_NUM_INTERP, GSM_LST_TOL, GSM_LST_MAXITER, GSM_LST_WEIGHT**
  The coordinate machinery; see the corresponding FSM options.
  TYPE: Double, Integer, Integer, Double, Integer, Double
  DEFAULT: 7.0E-3, 8, 100, 1.0E-3, 500, 1.0E-2

- **GSM_DQMAX, GSM_DQMIN**
  Single-ended only: the longest and shortest driving step, as displacements in the internal coordinate set. The step taken is interpolated between them by how far the drivers say there is left to go, so it is long at the start and short near the barrier. GSM_DQMIN at or below zero asks for a quarter of GSM_DQMAX, which is the reference implementation's rule. Raising GSM_DQMAX makes each node cover more ground and is what to try when the string runs out of nodes before it crosses a barrier.
  TYPE: Double, Double
  DEFAULT: 0.8, 0.0
  OPTIONS: n User-defined value.

- **GSM_SE_OSTEPS, GSM_SE_TOL**
  Single-ended only: how many steps the frontier node may take before the next one is grown from it, and the rms perpendicular gradient in hartree Å^{-1} at which it stops early. Both are far tighter than what a double-ended node gets, and they have to be: the next node is placed by stepping from this one, so any strain left here is carried forward and the string climbs a wall instead of a barrier. Loosening GSM_SE_TOL is the first thing that will appear to make a run cheaper and the first thing that will make it wrong.
  TYPE: Integer, Double
  DEFAULT: 30, 5.0E-3
  OPTIONS: n User-defined value.

- **GSM_HESS, GSM_TS_SEARCH**
  Build the approximate Hessian at the highest node and follow the string with a P-RFO saddle search, exactly as in Section 9.3. With the climbing image the search usually has little left to do.
  TYPE: Boolean, Boolean
  DEFAULT: TRUE, TRUE
  OPTIONS: TRUE, FALSE

9.14 NEB Module Options

- **NEB_PRODUCT_FILE**
  An xyz file holding the product structure, in angstrom, with the atoms in the same order as the reactant's; it must be enclosed in double quotes. Left out, the product is read from a `$set_product` block.
  TYPE: Line
  DEFAULT: none

- **NEB_ATOM_MAP**
  As FSM_ATOM_MAP, for the band.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **NEB_NIMAGES**
  Movable images. The band carries these plus the two fixed endpoints. More images resolve the path better and cost one gradient each per iteration; 4–20 is the usual range.
  TYPE: Integer
  DEFAULT: 8
  OPTIONS: n User-defined value.

- **NEB_MAXITER**
  Iterations of the band. Every iteration costs one gradient per movable image.
  TYPE: Integer
  DEFAULT: 100
  OPTIONS: n User-defined value.

- **NEB_CLIMB**
  Let the highest image climb to the saddle point, which is the difference between CI-NEB and the plain method.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **NEB_CLIMB_START**
  Plain iterations before climbing begins. Climbing from the first iteration is unstable: the image would climb a direction the rest of the band has not yet resolved.
  TYPE: Integer
  DEFAULT: 5
  OPTIONS: n User-defined value.

- **NEB_NCLIMB**
  How many images climb. More than one is worth having only when the path has two maxima of comparable height.
  TYPE: Integer

DEFAULT: 1
OPTIONS: n User-defined value.

- **NEB_K**
  The spring constant, hartree bohr^{-2}. Because of the force projection the converged band is insensitive to it over orders of magnitude; it only decides how stiffly the images are held apart along the path.
  TYPE: Double
  DEFAULT: 0.05
  OPTIONS: n User-defined value.

- **NEB_VARIABLE_K, NEB_K_MIN**
  Variable spring constants, scaled linearly with the image energy between NEB_K at the top of the barrier and NEB_K_MIN at the reference energy or below, the reference being the higher of the two endpoints [48]. This packs the images more closely where the reaction coordinate matters and lets them spread out over the flat regions.
  TYPE: Boolean, Double
  DEFAULT: FALSE, 0.01
  OPTIONS: TRUE, FALSE

- **NEB_OPT**
  How the band is moved.
  TYPE: String
  DEFAULT: QM
  OPTIONS: QM, SD

  - QM : quick-min, the projected velocity Verlet — each image keeps a velocity and discards whatever part of it no longer points along the force.
  - SD : steepest descent, no memory at all.

- **NEB_DT, NEB_MAX_DISP**
  The time step of the optimizer, and a ceiling in angstrom on how far any one coordinate may move in a single step. The whole displacement is scaled rather than clipped component by component, so that its direction survives.
  TYPE: Double
  DEFAULT: 0.5, 0.1
  OPTIONS: n User-defined value.

- **NEB_TOL_MAXF, NEB_TOL_RMSF**
  Convergence: the largest and the rms component of the force acting on the movable images, in hartree bohr^{-1}.
  TYPE: Double
  DEFAULT: 3.0E-3, 1.0E-3
  OPTIONS: n User-defined value.

- **NEB_BAND_FILE**
  An xyz file holding a path — one structure after another, in the format the freezing

string and NEB itself write — to start the band from instead of interpolating one. It must be enclosed in double quotes. The endpoints stay the ones the input gives; the file supplies the shape between them, resampled at equal arclength to NEB_NIMAGES images. Left out, which is the default, the band is interpolated.

This exists for the case where no interpolation between the two endpoints is a physically possible path — the collinear HCN $\rightleftharpoons$ HNC trap described in Section 9.7 is the example — and there is no way round it but to supply the path. It is worth saying what it is *not* for: seeding the band with the freezing string's own output, which sounds like it should pay and does not. On $SiH_2 + H_2 \rightleftharpoons SiH_4$ the seeded band does start better, reaching $|F|_{\max} = 0.014$ at the fifth iteration where the interpolated one is still at 0.044, but the cost of a band is in the tail of the convergence and not in its first steps: 29 iterations against 33, and once the 31 gradients the string itself cost are counted, 265 against 266. A wash.
TYPE: Line
DEFAULT: none

- **NEB_INTERP**
  How the initial band is built — LST, a straight Cartesian line, or a straight line in internal coordinates. Only the starting path is concerned; the band is optimized in Cartesians afterwards either way. See Section 9.3 for what separates the three.
  TYPE: String
  DEFAULT: LST
  OPTIONS: LST, CART, RIC

- **NEB_NUM_INTERP, NEB_LST_TOL, NEB_LST_MAXITER, NEB_LST_WEIGHT**
  The LST machinery of the initial band; see the corresponding FSM options.
  TYPE: Integer, Double, Integer, Double
  DEFAULT: 100, 1.0E-3, 500, 1.0E-2

- **NEB_RIC_TOL, NEB_RIC_MAXITER**
  The RIC machinery of the initial band; see the corresponding FSM options.
  TYPE: Double, Integer
  DEFAULT: 7.0E-3, 8

- **NEB_HESS, NEB_TS_SEARCH**
  Build the approximate Hessian at the highest image and follow the band with a P-RFO saddle search, exactly as in Section 9.3. With the climbing image the search usually has nothing left to do.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

9.15 IRC Module Options

- **READ_CART_HESS**

Do read Cartesian Hessian? For `STARTING_POINT=TS`, the Hessian is required.
TYPE: Boolean
DEFAULT: TRUE
OPTIONS: TRUE, FALSE

- **STARTING_POINT**
  Type of starting point for the IRC computation.
  TYPE: String
  DEFAULT: TS
  OPTIONS: TS, NONTS
  - TS : Start from a transition-state structure and determine the initial direction from the selected Hessian eigenvector.
  - NONTS : Start from the current geometry and use the gradient information directly.

- **DIRECTION**
  Direction of the IRC path.
  TYPE: String
  DEFAULT: FORWARD
  OPTIONS: FORWARD, REVERSE, BOTH
  - FORWARD : Follow the IRC path in the forward direction.
  - REVERSE : Follow the IRC path in the reverse direction.
  - BOTH : Perform forward and reverse IRC computations in sequence.

- **IRC_ALGORITHM**
  IRC propagation algorithm. `HPC` falls back on `SGD`, with a warning, when the run has no Hessian at a transition state to start from.
  TYPE: String
  DEFAULT: HPC
  OPTIONS: HPC, SGD
  - SGD : Stabilized Euler with a line search along the gradient bisector (Ishida–Morokuma–Komornicki, Schmidt–Gordon–Dupuis). Gradients only, two gradients and one energy per point.
  - HPC : Hessian based predictor–corrector (Hratchian–Schlegel), the integrator used by Gaussian. One gradient per point and a more accurate path; needs `STARTING_POINT=TS` and a Hessian file.

- **IRC_HESS_UPDATE**
  Hessian updating formula used along the path by `IRC_ALGORITHM=HPC`. The Hessian is computed analytically only at the transition state.
  TYPE: String
  DEFAULT: BOFILL
  OPTIONS: BOFILL, MS, PSB

  - BOFILL : A mixture of MS and PSB weighted by how well conditioned the MS denominator is; the one recommended by Hratchian and Schlegel for path following.
  - MS : Murtagh–Sargent, the symmetric rank one formula.
  - PSB : Powell-symmetric-Broyden.

- **FOLLOW_ROOT**
  Hessian eigenvector to follow for `STARTING_POINT=TS`.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **NUM_STEP**
  Maximum number of IRC steps to perform in a given direction.
  TYPE: Integer
  DEFAULT: 6
  OPTIONS: n User-defined value.

- **IRC_MAXITER**
  Maximum number of IRC iterations.
  TYPE: Integer
  DEFAULT: 100
  OPTIONS: n User-defined value.

- **PRINT_LEVEL**
  The amount of information to print to the output file.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **IRC_STEP_SIZE**
  Trial step length for the line search along the gradient bisector, in Bohr·amu$^{1/2}$. With the default of 0.0 the length is chosen from `ALPHA` and the angle between the two gradients, which is the recommended setting.
  TYPE: Double
  DEFAULT: 0.0
  OPTIONS: n User-defined value.

- **ALPHA**
  Length of the steepest-descent step of an IRC step, in Bohr·amu$^{1/2}$; it sets how closely the points of the path are spaced.
  TYPE: Double
  DEFAULT: 0.15
  OPTIONS: n User-defined value.

- **MAX_STEP**
  Maximum step size in Bohr.

TYPE: Double
DEFAULT: 0.5
OPTIONS: n User-defined value.

- **TOL_MAXG**
  Convergence criterion for IRC: maximum gradient (cartesian coordinates, atomic units).
  TYPE: Double
  DEFAULT: 4.5e-4
  OPTIONS: n User-defined value.

- **TOL_RMSG**
  Convergence criterion for IRC: rms gradient (cartesian coordinates, atomic units).
  TYPE: Double
  DEFAULT: 3.0e-4
  OPTIONS: n User-defined value.

- **TOL_DISP**
  Displacement tolerance used in the IRC procedure.
  TYPE: Double
  DEFAULT: 0.005
  OPTIONS: n User-defined value.

- **E_CONVERGENCE**
  Energy convergence threshold.
  TYPE: Double
  DEFAULT: 1.0e-8
  OPTIONS: n User-defined value.

- **CUTOFF**
  Cutoff value for numerical procedures.
  TYPE: Double
  DEFAULT: 1.0e-10
  OPTIONS: n User-defined value.

10 FREQ: Vibrational Frequencies

Code Authors: B. Ermiş and U. Bozkaya.

10.1 Overview

The `freq` module generates displaced geometries from the initial molecular geometry to calculate the harmonic and anharmonic vibrational frequencies of chemical structures [52].

By default, the `freq` module works in serial mode. However, the parallel mode is also available and recommended for high-cost computations. Each displaced geometry is formed by the infrared program to be a separate input file. The input file type is set according to the selected program (`e.g. software program_name`). The generated input files are run in the selected program for the gradient calculations. Because these calculations are independent of each other, they can be run simultaneously on different computers. Thus, `freq` enables one to compute infrared spectroscopy at a lower cost compared with the serial mode.

The package comprises:

- Displaced geometry
- Harmonic vibrational frequencies
- Anharmonic vibrational frequencies
- Thermodynamic analysis
- Infrared intensity
- The `freq` module creates `epsilon_harm.out` or `epsilon_anharm.out` file to visualize the infrared spectrum. It also creates a file in molden format to animate vibrational modes.

The parallel mode of the `freq` module can be integrated with several quantum chemistry software such as Psi4 [53], Qchem [54], Cfour [55], Molpro [56], and Orca [57].

Frequencies of a fragmented molecule

With `MOL_FRAG TRUE` and a fragment method that has an analytic gradient (`FRAG_METHOD LSSMF` or `MFCC`), a `JOBTYPE FREQ` run computes the fragment gradient at every displaced geometry, exactly as it computes any other analytic gradient there, and builds the Hessian from the central differences of those. The result is the harmonic spectrum of the fragment energy surface: it is consistent by construction, since the fragment gradient is the exact derivative of the fragment energy, and it carries the fragmentation error of that energy and nothing else. The `$set_frag` and `$set_lssmf` blocks are the ones a fragmented gradient or

optimization takes. IR intensities are not available – there is no dipole of the whole molecule in a fragment run – and are reported as zero.

Both the frequency module (`FREQ_NPROC`, the MPI ranks over the displacements) and the fragment method (`FRAG_NPROC`, the ranks over the fragments) can run in parallel, and only one of them does at a time: the workers are forked processes, MPI is not fork-safe, and two layers dividing one communicator would deadlock. **PARALLEL_LAYER** chooses. With `FREQ` (the default) the displacements are spread over the workers and the ranks, and each fragment gradient inside them runs its fragments one after another – `FRAG_NPROC` is set aside with a note. With `FRAG` the displacements run one after another on every rank, and each fragment gradient spreads its fragments as it would on its own – `FREQ_NPROC` is set aside. The displacements are all the same size and there are $6N$ of them, so the default balances perfectly and communicates once; the fragment layer is the choice when one fragment gradient is itself too large for one process, or when the memory of `FREQ_NPROC` concurrent whole-molecule gradients is the limit. The frequencies are the same either way, to the last digit.

```
$set_globals
   jobtype freq
   mol_frag true
   ...
$end_globals

$set_frag
   frag_method lssmf
   frag_jobtype grad
   mode run
$end_frag

$set_freq
   freq_nproc 8
   parallel_layer freq
$end_freq
```

10.2 Input Examples

*Harmonic vibrational frequency computation at the `mp2/cc-pvdz` level in serial mode:

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
```

```
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method mp2
  reference rhf
  jobtype freq
  freeze_core true
  step_bohr 0.005
$end_globals

$set_freq
  anharm_level 0
  derive_level 2
  mode run
  temperature 298.15
  pressure 101325.0
  point_group C1
  axis_n 1
  energy -76.22842455
$end_freq
```

*Anharmonic vibrational frequency computation at the `mp2/cc-pvdz` level in parallel mode is given below. By selecting `MODE SOW`, a separate file is created for each gradient computation. Make sure the method you choose has an analytic gradient.

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method mp2
  reference rhf
  jobtype freq
  freeze_core true
  step_bohr 0.005
$end_globals

$set_freq
  anharm_level 2
  derive_level 4
```

```
  mode sow
$end_freq
```

*When each gradient computation is completed, you can proceed to the frequency computation by selecting `MODE REAP`:

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
  jobtype freq
  step_bohr 0.005
$end_globals

$set_freq
  anharm_level 2
  derive_level 4
  mode reap
  temperature 298.15
  pressure 101325.0
  energy -76.22842455
$end_freq
```

*Anharmonic vibrational frequency computation at the `mp2/cc-pvdz` level in parallel mode is given below. In this example the `PSI4` software is employed instead of `MacroQC`. When an external software is used, the options for the external software should be provided in the `DG_EXTERNAL` block. `MacroQC` generates all necessary input files for the allowed external software. By selecting `MODE SOW`, a separate file is created for each gradient computation. Make sure the method you choose has an analytic gradient. When each gradient computation is completed, you can proceed to the frequency computation by selecting `MODE REAP`.

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
  jobtype freq
  step_bohr 0.005
$end_globals
```

```
$set_freq
  anharm_level 2
  derive_level 4
  mode sow
  software psi4
$end_freq

$SET_DG_EXTERNAL
set {
basis cc-pvdz
df_basis_scf cc-pvdz-jkfit
df_basis_cc cc-pvdz-ri
reference rhf
scf_type df
mp_type df
freeze_core true
qc_module occ
oeprop true
}
gradient('mp2')
$END_DG_EXTERNAL
```

A frequency calculation of a fragmented molecule (`MOL_FRAG TRUE` together with `JOBTYPE FREQ`) is refused. A Hessian of a fragmented molecule has to be assembled the way the fragment gradient is — each fragment's second derivative carried into the parent with that fragment's own coefficient, caps and embedding charges — and that is not implemented. Run the frequency without `MOL_FRAG`.

10.3 FREQ Module Options

- **MODE**
  Type of the mode.
  TYPE: String
  DEFAULT: RUN
  OPTIONS: RUN, SOW, REAP

  - RUN: The displaced geometries are computed by this run rather than written out for another one. Whether they are done one after another or in parallel is set by `FREQ_NPROC` and, across nodes, by MPI — see section 5.4; `MODE` says who does the work and not how many of them there are.

    This value used to be spelled `SERIAL`, which named an execution style rather than a destination. `SERIAL` is still accepted, so existing input files need no change.
  - SOW: Creates perturbed geometries and allows you to run them in parallel by generating input files for displaced geometries.

  - REAP: When analytic gradient computations are completed for displaced geometries, `REAP` option requests to compute vibrational frequencies. Note that when the `MacroQC` program is executed with `REAP` option, all outputs for displaced geometries should be available in the working directory.

- **FREQ_NPROC**
  How many displaced geometries to compute at the same time. A frequency calculation spends nearly all of its time on gradients at displaced geometries, and those gradients do not depend on one another, so they can be run concurrently: each worker is a separate process with its own molecule, options and scratch files, and computes a share of the displacements. The number is a real count of simultaneous electronic structure calculations, so it should not exceed the cores available, and memory scales with it — the `MEMORY` option is per process. Where the gradients themselves are threaded (through the BLAS, for instance), the two levels share one budget: `FREQ_NPROC` $\times$ threads per worker $\leq$ cores.

  It applies to `MODE RUN` only; `SOW` and `REAP` write files or read them, and there is nothing to spread. Across machines it is MPI that divides the displacements, and the two compose — one rank per node, `FREQ_NPROC` set to that node's cores. The answers do not depend on either: the same frequencies come out to the last digit whatever the worker and rank count.
  TYPE: Integer
  DEFAULT: 1

- **PARALLEL_LAYER**
  For the frequency of a fragmented molecule (`MOL_FRAG`): which of the two parallel layers runs. `FREQ` spreads the displacements (`FREQ_NPROC`, the MPI ranks) and runs the fragments of each gradient one after another; `FRAG` runs the displacements one after another and lets each fragment gradient spread its fragments (`FRAG_NPROC`, the ranks). Never both. Without `MOL_FRAG` it has no effect.
  TYPE: String
  DEFAULT: FREQ
  OPTIONS: FREQ, FRAG

- **SOFTWARE**
  Which quantum chemistry software to use?
  TYPE: String
  DEFAULT: MACROQC
  OPTIONS: MACROQC, PSI4, QCHEM, MOLPRO, ORCA, CFOUR

- **DERIVE_LEVEL**
  Specifies the level of energy derivatives.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: 2, 3, 4

- **ANHARM_LEVEL**
  Do compute anharmonic frequencies?
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: 0, 1, 2
  - 0: Harmonic vibration frequency.
  - 1: Anharmonic correttion (cubic terms).
  - 2: Anharmonic correttion (quartic terms).

- **PRINT_LEVEL**
  The amount of information to print to the output file.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **TOL_ANHARM**
  Tolerance value for Fermi resonance [58].
  TYPE: Double
  DEFAULT: 200.0
  OPTIONS: n User-defined value.

- **TOL_MARTIN**
  Tolerance value for criterion suggested by Martin and co-workers [59].
  TYPE: Double
  DEFAULT: 10.0
  OPTIONS: n User-defined value.

- **W_LORENTZ**
  Lorentz width at half height.
  TYPE: Double
  DEFAULT: 10.0
  OPTIONS: n User-defined value.

- **PRESSURE**
  Pressure (in Pascal).
  TYPE: Double
  DEFAULT: 101325.0
  OPTIONS: n User-defined value.

- **TEMPERATURE**
  Temperature (in Kelvin).
  TYPE: Double
  DEFAULT: 298.15
  OPTIONS: n User-defined value.

- **ENERGY**
  Electronic energy at the optimized geometry (in atomic units).
  TYPE: Double
  DEFAULT: 298.15
  OPTIONS: n User-defined value.

- **POINT_GROUP**
  Full point group.
  TYPE: String
  DEFAULT: C1
  OPTIONS: ATOM, C1, CI, CS, C_INF_V, D_INF_H, TD, OH, IH, CN, CNV, CNH, DN, DND, DNH, SN

- **AXIS_N**
  Point group axis, which is needed to determine rotational symmetry number. Users need to provide this option in the case of CN, CNV, CNH, DN, DND, DNH, and SN point groups. In the case of other point groups, it is unnecessary to set this option.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value. For CN, CNV, CNH, DN, DND, DNH, and SN point groups n are equal to N of the point group. For example for C2V group $n = 2$.

11 DFOCC : Orbital-Optimized Coupled-Cluster and Moller–Plesset Perturbation Theories

Code Authors: U. Bozkaya with contributions from A. Ünal and Y. Alagöz.

11.1 Theory

What follows is a very basic description of orbital-optimized Moller–Plesset perturbation theory as implemented in MACROQC.

The orbital variations may be expressed by means of an exponential unitary operator

$$\widetilde{\hat{p}}^{\dagger} = e^{\hat{K}} \hat{p}^{\dagger} e^{-\hat{K}} \tag{14}$$

$$\widetilde{\hat{p}} = e^{\hat{K}}\ \hat{p}\ e^{-\hat{K}} \tag{15}$$

$$|\widetilde{p}\rangle = e^{\hat{K}}\ |p\rangle \tag{16}$$

where $\hat{K}$ is the orbital rotation operator

$$\hat{K} = \sum_{p,q} K_{pq}\ \hat{E}_{pq} = \sum_{p>q} \kappa_{pq}\ \hat{E}^{-}_{pq} \tag{17}$$

$$\hat{E}_{pq} = \hat{p}^{\dagger}\hat{q} \tag{18}$$

$$\hat{E}^{-}_{pq} = \hat{E}_{pq}\ -\ \hat{E}_{qp} \tag{19}$$

$$\mathbf{K} = Skew(\kappa) \tag{20}$$

The effect of the orbital rotations on the MO coefficients can be written as

$$\mathbf{C}(\kappa) = \mathbf{C}^{(\mathbf{0})}\ e^{\mathbf{K}} \tag{21}$$

where $\mathbf{C}^{(\mathbf{0})}$ is the initial MO coefficient matrix and $\mathbf{C}(\kappa)$ is the new MO coefficient matrix as a function of κ. Now, let us define a variational energy functional (Lagrangian) as a function of κ,

OMP2 [60, 61, 11, 10, 12]:

$$\begin{aligned}\widetilde{E}(\boldsymbol{\kappa}) &= \langle 0|\hat{H}^{\kappa}|0\rangle\ +\ \langle 0|\big(\hat{W}^{\kappa}_{N}\hat{T}^{(1)}_{2}\big)_{c}|0\rangle \\ &+\ \langle 0|\big\{\hat{\Lambda}^{(1)}_{2}\ \big(\hat{f}^{\kappa}_{N}\hat{T}^{(1)}_{2}\ +\ \hat{W}^{\kappa}_{N}\big)_{c}\big\}_{c}|0\rangle,\end{aligned} \tag{22}$$

OMP3 [62, 63, 64, 13, 18]:

$$\begin{aligned}\widetilde{E}(\boldsymbol{\kappa}) &= \langle 0|\hat{H}^{\kappa}|0\rangle \\ &+\ \langle 0|\big(\hat{W}^{\kappa}_{N}\hat{T}^{(1)}_{2}\big)_{c}|0\rangle\ +\ \langle 0|\big(\hat{W}^{\kappa}_{N}\hat{T}^{(2)}_{2}\big)_{c}|0\rangle \\ &+\ \langle 0|\{\hat{\Lambda}^{(1)}_{2}\ \big(\hat{f}^{\kappa}_{N}\hat{T}^{(1)}_{2}\ +\ \hat{W}^{\kappa}_{N}\big)_{c}\}_{c}|0\rangle \\ &+\ \langle 0|\{\hat{\Lambda}^{(1)}_{2}\ \big(\hat{f}^{\kappa}_{N}\hat{T}^{(2)}_{2}\ +\ \hat{W}^{\kappa}_{N}\hat{T}^{(1)}_{2}\big)_{c}\}_{c}|0\rangle \\ &+\ \langle 0|\{\hat{\Lambda}^{(2)}_{2}\ \big(\hat{f}^{\kappa}_{N}\hat{T}^{(1)}_{2}\ +\ \hat{W}^{\kappa}_{N}\big)_{c}\}_{c}|0\rangle,\end{aligned} \tag{23}$$

OMP2.5 [65, 66, 13]:

$$\begin{aligned}\widetilde{E}(\boldsymbol{\kappa}) &= \langle 0|\hat{H}|0\rangle \\ &+ \langle 0|\big(\hat{W}_N^\kappa \hat{T}_2^{(1)}\big)_c|0\rangle \;+\; \frac{1}{2}\langle 0|\big(\hat{W}_N^\kappa \hat{T}_2^{(2)}\big)_c|0\rangle \\ &+ \langle 0|\{\hat{\Lambda}_2^{(1)}\;\big(\hat{f}_N^\kappa \hat{T}_2^{(1)} \;+\; \hat{W}_N^\kappa\big)_c\}_c|0\rangle \\ &+ \frac{1}{2}\langle 0|\{\hat{\Lambda}_2^{(1)}\;\big(\hat{f}_N^\kappa \hat{T}_2^{(2)} \;+\; \hat{W}_N^\kappa \hat{T}_2^{(1)}\big)_c\}_c|0\rangle \\ &+ \frac{1}{2}\langle 0|\{\hat{\Lambda}_2^{(2)}\;\big(\hat{f}_N^\kappa \hat{T}_2^{(1)} \;+\; \hat{W}_N^\kappa\big)_c\}_c|0\rangle, \end{aligned} \tag{24}$$

OLCCD [67, 68, 14]:

$$\begin{aligned}\widetilde{E}(\boldsymbol{\kappa}) &= \langle 0|\hat{H}^\kappa|0\rangle \;+\; \langle 0|\big(\hat{W}_N^\kappa \hat{T}_2\big)_c|0\rangle \\ &+ \langle 0|\{\hat{\Lambda}_2\;\big(\hat{W}_N^\kappa \;+\; \hat{H}_N^\kappa \hat{T}_2\big)_c\}_c|0\rangle, \end{aligned} \tag{25}$$

OCCD [60, 69]:

$$\widetilde{E}(\boldsymbol{\kappa}) \;=\; \langle 0|(1+\hat{\Lambda}_2)e^{-\hat{T}_2}\hat{H}^\kappa e^{\hat{T}_2}|0\rangle, \tag{26}$$

where subscript c means only connected diagrams are included, $\hat{T}_2$ and $\hat{\Lambda}_2$ are cluster double excitation and de-excitation operators, respectfully, and $\hat{H}^\kappa$, $\hat{f}_N^\kappa$, $\hat{W}_N^\kappa$, and $\hat{H}_N^\kappa$ are defined as

$$\hat{H}^\kappa \;=\; e^{-\hat{K}}\hat{H}e^{\hat{K}}, \tag{27}$$

$$\hat{f}_N^\kappa \;=\; e^{-\hat{K}}\hat{f}_N^d e^{\hat{K}}, \tag{28}$$

$$\hat{W}_N^\kappa \;=\; e^{-\hat{K}}\hat{W}_N e^{\hat{K}}, \tag{29}$$

$$\hat{H}_N^\kappa \;=\; e^{-\hat{K}}\hat{H}_N e^{\hat{K}}. \tag{30}$$

The energy gradient and Hessian can be written as follows:

$$w_{pq} \;=\; \left.\frac{\partial \widetilde{E}}{\partial \kappa_{pq}}\right|_{\boldsymbol{\kappa}=0}, \tag{31}$$

$$A_{pq,rs} \;=\; \left.\frac{\partial^2 \widetilde{E}}{\partial \kappa_{pq}\partial \kappa_{rs}}\right|_{\boldsymbol{\kappa}=0}, \tag{32}$$

Then the energy can be expanded up to second-order as follows:

$$\widetilde{E}^{(2)}(\boldsymbol{\kappa}) = \widetilde{E}^{(0)} + \boldsymbol{\kappa}^\dagger w + \frac{1}{2}\,\boldsymbol{\kappa}^\dagger A\kappa, \tag{33}$$

where $\boldsymbol{\kappa}$ is the MO rotation vector, $\boldsymbol{w}$ is the MO gradient vector, and $\boldsymbol{A}$ is the MO Hessian matrix. Hence, minimizing the energy with respect to $\boldsymbol{\kappa}$ yields

$$\boldsymbol{\kappa} \;=\; -\boldsymbol{A}^{-1}w. \tag{34}$$

Table 7: OO methods.

Name	Calls Method	Energy	Gradient
`omp2`	Density-Fitted Orbital-Optimized MP2	RHF/UHF/ROHF	RHF/UHF/ROHF
`omp3`	Density-Fitted Orbital-Optimized MP3	RHF/UHF/ROHF	RHF/UHF/ROHF
`omp2.5`	Density-Fitted Orbital-Optimized MP2.5	RHF/UHF/ROHF	RHF/UHF/ROHF
`olccd`	Density-Fitted Orbital-Optimized LCCD	RHF/UHF/ROHF	RHF/UHF/ROHF
`occd`	Density-Fitted Orbital-Optimized CCD	RHF/UHF/ROHF	RHF/UHF/ROHF
`occd(t)`	Density-Fitted Orbital-Optimized CCD(T)	RHF/UHF/ROHF	NA
`occd(at)`	Density-Fitted Orbital-Optimized CCD(AT)	RHF/UHF/ROHF	NA

This final equation corresponds to the usual Newton-Raphson step.

The orbital-optimized MPn and CC methods currently supported in `dfocc` are outlined in Table 7.

Publications resulting from the use of the orbital-optimized code should cite the following publications:

OMP2 Bozkaya and co-workers [60, 61, 11, 10, 12].

OMP3 Bozkaya and co-workers [62, 63, 64, 13, 18].

OMP2.5 Bozkaya and co-workers [65, 66, 13].

OLCCD Bozkaya and co-workers [67, 68, 14].

OCCD Bozkaya and co-workers [69].

11.2 Conventional (Non-OO) Coupled-Cluster and Moller-Plesset Perturbation Theories

Non-orbital-optimized counterparts to MPn and CC methods currently supported in `dfocc` are outlined in Table 8.

Table 8: Non-OO methods.

Name	Calls Method	Energy	Gradient
`mp2`	Density-Fitted MP2	RHF/UHF	RHF/UHF
`mp3`	Density-Fitted MP3	RHF/UHF	RHF/UHF
`mp2.5`	Density-Fitted MP2.5	RHF/UHF	RHF/UHF
`lccd`	Density-Fitted LCCD	RHF/UHF	RHF/UHF
`ccd`	Density-Fitted CCD	RHF/UHF	RHF/UHF
`ccsd`	Density-Fitted CCSD	RHF/UHF	RHF/UHF
`ccsd(t)`	Density-Fitted CCSD(T)	RHF/UHF	RHF/UHF
`ccsd(at)`	Density-Fitted CCSD(AT)	RHF/UHF	NA

Publications resulting from the use of the MPn and CC codes should cite the following publications:

MP2 Bozkaya and co-workers [60, 61, 11, 10, 12].

MP3 Bozkaya and co-workers [62, 63, 64, 13, 18].

MP2.5 Bozkaya and co-workers [65, 66, 13].

LCCD Bozkaya and co-workers [67, 68, 14].

LCCD Bozkaya and co-workers [67, 68, 14].

CCD Bozkaya and co-workers [60, 15, 16, 69].

CCSD Bozkaya and co-workers [60, 15, 16, 69].

CCSD(T) Bozkaya and co-workers [70, 15, 17].

CCSD(AT) Bozkaya and co-workers [70, 15].

11.3 Input Examples

*Single point `omp2/cc-pvdz` computation:

```
$set_molecule H2O
  0 1
  O   0.00000   0.00000  -0.06577
  H   0.00000  -0.75906   0.52195
  H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method omp2
  reference rhf
  jobtype energy
  freeze_core true
$end_globals

$set_dfocc
  e_convergence 1e-8
  r_convergence 1e-6
  mo_maxiter 50
  max_mograd_convergence 1.0e-3
  rms_mograd_convergence 1.0e-6
$end_dfocc
```

*Single point `ccsd(t)/cc-pvdz` computation:

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
```

```
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    aux_basis_corr cc-pvdz-ri
    method ccsd(t)
    reference rhf
    jobtype energy
    freeze_core true
$end_globals

$set_dfocc
    e_convergence 1e-8
    r_convergence 1e-6
$end_dfocc
```

*Single point `fno-ccsd(t)/cc-pvdz` computation:

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    aux_basis_corr cc-pvdz-ri
    method ccsd(t)
    reference rhf
    jobtype energy
    freeze_core true
$end_globals

$set_dfocc
    e_convergence 1e-8
    r_convergence 1e-6
    nat_orbs true
    occ_tolerance 1.0e-5
$end_dfocc
```

*Ionization potentials via the extended Koopmans' theorem (EKT) at the `ccsd(t)/cc-pvdz` level:

```
$set_molecule H2O
```

```
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    aux_basis_corr cc-pvdz-ri
    method ccsd(t)
    reference rhf
    jobtype grad
    freeze_core true
$end_globals

$set_dfocc
    e_convergence 1e-8
    r_convergence 1e-6
    ekt_ip true
$end_dfocc
```

*Frozen-natural orbitals at the `ccsd(t)/cc-pvdz` level:

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    aux_basis_corr cc-pvdz-ri
    method ccsd(t)
    reference rhf
    jobtype energy
    freeze_core true
$end_globals

$set_dfocc
    e_convergence 1e-8
    r_convergence 1e-6
    nat_orbs true
    occ_tolerance 1e-5
```

```
$end_dfocc
```

*Cholesky-decomposed (CD) `ccsd/cc-pvdz` energy. The ERI tensor is factorized on the fly, so no correlation auxiliary basis is used; the accuracy is controlled by a single threshold instead of the quality of a fitting basis. CD is available for ENERGIES only (no analytic gradients), and is switched on the same way for every method of this module, orbital-optimized ones included (`mp2`, `mp2.5`, `mp3`, `lccd`, `ccd`, `ccsd`, `ccsd(t)`, `ccsd(at)`, `omp2`, `omp2.5`, `omp3`, `olccd`, `occd`, `occd(t)`, `occd(at)`):

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    method ccsd
    reference rhf
    jobtype energy
    freeze_core true
$end_globals

$set_dfocc
    cholesky true
    cholesky_tolerance 1e-4
    e_convergence 1e-8
    r_convergence 1e-6
$end_dfocc
```

11.4 DFOCC Module Options

- **CHOLESKY**
  Do Cholesky decomposition (CD) of the ERI tensor instead of density fitting? With `true`, the two-electron integrals are factorized as $(\mu\nu|\lambda\sigma) \approx \sum_Q L_Q(\mu\nu)L_Q(\lambda\sigma)$, the factors replace the DF $B(Q|\mu\nu)$ everywhere, and no correlation auxiliary basis (`aux_basis_corr`) is needed. Available for ENERGIES of every DFOCC method – MP2, MP2.5, MP3, LCCD, CCD, CCSD, CCSD(T), CCSD(AT) and their orbital-optimized counterparts OMP2, OMP2.5, OMP3, OLCCD, OCCD, OCCD(T), OCCD(AT). Analytic gradients require DF. In the orbital-optimized methods the decomposition is performed ONCE: the Cholesky factors live in the AO basis and do not depend on the orbitals, so only the MO transformation is repeated in the orbital loop.
  TYPE: String

DEFAULT: FALSE
OPTIONS: TRUE, FALSE

- **CHOLESKY_TOLERANCE**
  Threshold of the CD. The decomposition stops when the largest remaining diagonal of the residual matrix drops below this value, which also bounds every reconstructed integral error, $|(\mu\nu|\lambda\sigma) - \sum_Q L_Q L_Q| \leq \delta$. Smaller values give more Cholesky vectors and an energy closer to the exact-ERI (conventional) result.

  The SCF may be run with CD independently (`SCF/cholesky`), and the two choices are free to differ: CD, DF or conventional SCF combines with DF or CD here. When BOTH sides use CD at the SAME threshold, this module reuses the factors the SCF already built instead of decomposing again – they are AO-basis quantities and do not depend on the orbitals. Every other combination builds what it needs from scratch.
  TYPE: Double
  DEFAULT: 1.0e-4
  OPTIONS: n User-defined tolerance value.

- **DAVIDSON_TOLERANCE**
  Convergence criterion for Davidson (residuals).
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined tolerance value.

- **E3_SCALE**
  Scaling value for 3rd order energy correction (S. Grimme, Vol. 24, pp. 1529, J. Comput. Chem.)
  TYPE: Double
  DEFAULT: 0.25
  OPTIONS: n User-defined value.

- **E_CONVERGENCE**
  Convergence criterion for energy.
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined value.

- **LEVEL_SHIFT**
  Level shift to aid convergence.
  TYPE: Double
  DEFAULT: 0.02
  OPTIONS: n User-defined value.

- **MAX_MOGRAD_CONVERGENCE**
  Convergence criterion for maximum orbital gradient.
  TYPE: Double

DEFAULT: 1.0e-3
OPTIONS: n User-defined value.

- **MO_STEP_MAX**
  Maximum step size in orbital-optimization procedure.
  TYPE: Double
  DEFAULT: 0.5
  OPTIONS: n User-defined value.

- **MP2_OS_SCALE**
  MP2 opposite-spin scaling value.
  TYPE: Double
  DEFAULT: 6.0/5.0
  OPTIONS: n User-defined value.

- **MP2_SOS_SCALE**
  MP2 Spin-opposite scaling (SOS) value.
  TYPE: Double
  DEFAULT: 1.3
  OPTIONS: n User-defined value.

- **MP2_SOS_SCALE2**
  Spin-opposite scaling (SOS) value for optimized-MP2 orbitals.
  TYPE: Double
  DEFAULT: 1.2
  OPTIONS: n User-defined value.

- **MP2_SS_SCALE**
  MP2 same-spin scaling value.
  TYPE: Double
  DEFAULT: 1.0/3.0
  OPTIONS: n User-defined value.

- **OCC_PERCENTAGE**
  Cutoff for the occupation of MP2 virtual NOs in FNO-CCSD/CCSD(T). The number of virtual NOs is chosen so the occupation of the truncated virtual space is OCC_PERCENTAGE percent of occupation of the original MP2 virtual space. This option is only used if NAT_ORBS = true. This keyword overrides OCC_TOLERANCE.
  TYPE: Double
  DEFAULT: 99.0
  OPTIONS: n User-defined value.

- **OCC_TOLERANCE**
  Cutoff for the occupation of MP2 virtual NOs in FNO-CCSD/CCSD(T). Virtual NOs with occupations less than OCC_TOLERANCE will be discarded. This option is only used if NAT_ORBS = true.

TYPE: Double
DEFAULT: 1.0e-5
OPTIONS: n User-defined tolerance value.

- **OO_SCALE**
  OO scaling factor used in MSD.
  TYPE: Double
  DEFAULT: 0.01
  OPTIONS: n User-defined value.

- **PCG_CONVERGENCE**
  Convergence criterion for the residual vector of the preconditioned conjugate gradient method.
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined value.

- **SCG_TOLERANCE**
  Tolerance for strictly canonical.
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined tolerance value.

- **R_CONVERGENCE**
  Convergence criterion for amplitudes (residuals).
  TYPE: Double
  DEFAULT: 1.0e-5
  OPTIONS: n User-defined value.

- **REG_PARAM**
  Regularization parameter. Works with "REG_TYPE NORMAL" option.
  TYPE: Double
  DEFAULT: 0.4
  OPTIONS: n User-defined value.

- **KAPPA_REG_PARAM**
  Kappa-regularization parameter. Works with "REG_TYPE KAPPA" option.
  TYPE: Double
  DEFAULT: 1.1
  OPTIONS: n User-defined value.

- **RMS_MOGRAD_CONVERGENCE**
  Convergence criterion for RMS orbital gradient. Default adjusts depending on E_CONVERGENCE.
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined value.

- **ACTIVE_NAT_ORBS**
  An array containing the number of virtual natural orbitals per irrep (in Cotton order) so a user can specify the number of retained natural orbitals rather than determining them with OCC_TOLERANCE. This keyword overrides OCC_TOLERANCE and OCC_PERCENTAGE.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: n User-defined value.

- **CC_DIIS_MAX_VECS**
  Maximum number of vectors used in amplitude DIIS.
  TYPE: Integer
  DEFAULT: 6
  OPTIONS: n User-defined value.

- **CC_DIIS_MIN_VECS**
  Minimum number of vectors used in amplitude DIIS.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: n User-defined value.

- **CC_MAXITER**
  The Maximum number of iterations to determine the amplitudes. This option is also used for CIS iterations.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **CUTOFF**
  The cutoff value for numerical procedures.
  TYPE: Integer
  DEFAULT: 8
  OPTIONS: n User-defined value.

- **FOLLOW_ROOT**
  Follow QDPT root.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **INTEGRAL_CUTOFF**
  Cutoff value for DF integrals.
  TYPE: Integer
  DEFAULT: 9
  OPTIONS: n User-defined value.

- **MO_DIIS_NUM_VECS**
  The number of vectors used in orbital DIIS.
  TYPE: Integer
  DEFAULT: 6
  OPTIONS: n User-defined value.

- **MO_MAXITER**
  Maximum number of iterations to determine the orbitals.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **NUM_ROOTS**
  Number of CI Roots interested.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **PCG_MAXITER**
  Maximum number of preconditioned conjugate gradient iterations.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **PRINT_LEVEL**
  The amount of information to print to the output file.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **CC_LAMBDA**
  Do solve lambda amplitude equations?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE.

- **CIS_ALGORITHM**
  Algorithm used to build the CIS/TD-DFT sigma vector.
  TYPE: String
  DEFAULT: AO_BASIS
  OPTIONS: AO_BASIS, MO_BASIS

  – AO_BASIS: Keep the exchange in the AO basis, so the virtual–virtual three-index block b_Q^{AB} is never formed. The integrals used are selected with **AO_JK_TYPE**.

  – MO_BASIS: Contract the sigma vector over MO density-fitted integrals. This route always uses density fitting and requires b_Q^{AB}.

- **AO_JK_TYPE**
  Source of the Coulomb and exchange matrices for the AO-basis algorithm. The name states both which integrals are used (DF_ for density fitting, CONV_ERI_ for conventional four-index integrals) and how they are obtained (_DIRECT recomputed as needed, _INCORE built once and kept).
  TYPE: String
  DEFAULT: DF_DIRECT
  OPTIONS: DF_DIRECT, DF_INCORE, CONV_ERI_DIRECT, CONV_ERI_INCORE

  - DF_DIRECT: Three-index integrals are rebuilt on the fly for every sigma vector; the full $b_Q^{\mu\nu}$ tensor is never stored.
  - DF_INCORE: The stored $b_Q^{\mu\nu}$ is read from disk and reused.
  - CONV_ERI_DIRECT: Conventional four-index integrals, computed on the fly. The auxiliary basis is not used at all, so the excitation energies carry no fitting error. The quartet kernel is the one selected by **ERI_ALGORITHM** in the INTS block (Obara–Saika by default).
  - CONV_ERI_INCORE: The same conventional integrals, but built once and held in memory in packed form, after which each sigma vector is a single sweep over that array. This is normally the fastest choice, and is limited only by memory: $M(M+1)/2$ double-precision numbers with $M = n_{\mathrm{bf}}(n_{\mathrm{bf}}+1)/2$. The requirement is checked against **MEMORY** and reported before the integrals are built.

  Both conventional routes handle a non-symmetric transition density, so $\mathbf{K}[\mathbf{P}]$ and $\mathbf{K}[\mathbf{P}^T]$ – needed by the **A** and **B** blocks of the full TD-DFT (RPA) problem – are obtained from a single pass over the integrals. Range-separated hybrids (e.g. ωB97, ωB97X) are supported by every back-end except DF_DIRECT, which has no erf-attenuated three-index kernel; if no back-end was requested explicitly, such a run falls back to DF_INCORE and says so in the output. The conventional routes evaluate $\mathrm{erf}(\omega r_{12})/r_{12}$ exactly rather than through a one-sided fit, and are available only with the OS2 kernel.

- **CIS_GUESS_COEFF**
  Initial guess for unrestricted CIS (UCIS) computations.
  TYPE: String
  DEFAULT: MIX
  OPTIONS: MIX, SINGLET, TRIPLET

  - MIX: Use a mixture of singlet and triplet states.
  - SINGLET: Use singlet coefficients, which means $c_I^A = 1/\sqrt{2}$ and $c_i^a = 1/\sqrt{2}$.
  - TRIPLET: Use triplet coefficients, which means $c_I^A = 1/\sqrt{2}$ and $c_i^a = -1/\sqrt{2}$.

- **UNIT_GUESS_TYPE**
  Type of initial guesses.
  TYPE: String

DEFAULT: GEN
OPTIONS: LIMITED, GEN

- LIMITED: This option requests a limited guess to be generated around HOMO-LUMO orbitals. The dimension of guess is determined by the EXCITATION_RANGE option.
- GEN: All singly excited Slater determinants are used in generating an initial guess.

- **EXCITATION_RANGE**
  The gap between the occupied and virtual spaces from which initially excited state guesses are directed to the Davidson algorithm in CIS computations. The default value of 2 corresponds to HOMO-1, HOMO, LUMO, and LUMO+1.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: n User-defined value.

- **VECS_PER_ROOT**
  Subspace vectors used per root in CIS computations.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **SCHMIDT_ADD_RES_TOL**
  Tolerance value to add a guess vector of a root to the Davidson algorithm in CIS computations.
  TYPE: Double
  DEFAULT: 1.0e-3
  OPTIONS: n User-defined tolerance value.

- **CIS_E_CONV**
  Convergence criterion for excitation energy.
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined tolerance value.

- **CIS_RES_PRECONDITION_TOL**
  Tolerance value of CIS residual preconditioning.
  TYPE: Double
  DEFAULT: 1.0e-4
  OPTIONS: n User-defined tolerance value.

- **MAX_VECS**
  Maximum dimension of the subspace.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **COMPUT_S2**
  Do compute $\langle \hat{S}^2 \rangle$ for DF-OMP2/DF-MP2?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE.

- **DIAG_METHOD**
  Diagonalization method used for the CIS Hamiltonian.
  TYPE: String
  DEFAULT: DAVIDSON
  OPTIONS: DAVIDSON, FULL_DIAG

  - DAVIDSON: Use the Davidson algorithm, which is available for RHF and UHF references.
  - FULL_DIAG: Build the overall CIS Hamiltonian and diagonalize it. This option is available only for the RHF reference.

- **DO_DIIS**
  Do apply DIIS extrapolation?
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **DO_LEVEL_SHIFT**
  Do apply level shifting?
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **DO_SCS**
  Do perform spin-component-scaled OMP2 (SCS-OMP2)? In all computations, SCS-OMP2 energy is computed automatically. However, in order to perform geometry optimizations and frequency computations with SCS-OMP2, one needs to set 'DO_SCS' to true.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **DO_SOS???**
  Do perform spin-opposite-scaled OMP2 (SOS-OMP2)? In all computations, SOS-OMP2 energy is computed automatically. However, in order to perform geometry optimizations and frequency computations with SOS-OMP2, one needs to set 'DO_SOS' to true.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **EKT_IP**
  Do compute ionization potentials based on the extended Koopmans' theorem?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **MO_READ**
  Do read MO coefficients from external files?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **MO_WRITE**
  Do write MO coefficients to external files?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **HESS_TYPE**
  Type of the MO Hessian matrix.
  TYPE: String
  DEFAULT: HF
  OPTIONS: HF, APPROX_DIAG, APPROX_DIAG_EKT, APPROX_DIAG_HF

- **LINEQ_SOLVER**
  The solver will be used for simultaneous linear equations.
  TYPE: String
  DEFAULT: CDGESV
  OPTIONS: CDGESV, FLIN, POPLE

- **MP2_AMP_TYPE**
  The algorithm that is used to handle mp2 amplitudes. The DIRECT option means compute amplitudes on the fly whenever they are necessary.
  TYPE: String
  DEFAULT: DIRECT
  OPTIONS: DIRECT, CONV

- **NAT_ORBS**
  Do use MP2 NOs to truncate virtual space for CCD/CCSD and (T)?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **NBO**
  Do compute natural orbitals?
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **OCC_ORBS_PRINT**
  Do print OCC orbital energies?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **OEPROP**
  Do compute one electron properties?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **OPT_METHOD**
  The orbital optimization algorithm. Presently quasi-Newton-Raphson algorithm is available with several Hessian options.
  TYPE: String
  DEFAULT: QNR
  OPTIONS: QNR

- **ORB_RESP_SOLVER**
  The algorithm will be used for solving the orbital-response equations. The LINEQ option creates the MO Hessian and solves the simultaneous linear equations with the method chosen by the LINEQ_SOLVER option. The PCG option does not create the MO Hessian explicitly, instead, it solves the simultaneous equations iteratively with the preconditioned conjugate gradient method.
  TYPE: String
  DEFAULT: PCG
  OPTIONS: PCG, LINEQ

- **ORTH_TYPE**
  The algorithm for orthogonalization of MOs
  TYPE: String
  DEFAULT: MGS
  OPTIONS: MGS, GS

- **PCG_BETA_TYPE**
  Type of PCG beta parameter (Fletcher-Reeves or Polak-Ribiere).
  TYPE: String
  DEFAULT: FLETCHER_REEVES
  OPTIONS: FLETCHER_REEVES, POLAK_RIBIERE

- **PPL_TYPE**
  Type of the CCSD PPL term.
  TYPE: String
  DEFAULT: AUTO
  OPTIONS: AUTO, LOW_MEM, HIGH_MEM, CD

- **PRINT_CI_VECS**
  Do print CI coefficients in CIS computations?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **QCHF**
  Do perform a QCHF computation?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **REGULARIZATION**
  Do use regularized denominators?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **REG_TYPE**
  Specifies the regularization type for MP2 amplitudes. Works with "REGULARIZATION TRUE" option.
  TYPE: String
  DEFAULT: NORMAL
  OPTIONS: NORMAL, KAPPA

- **SCS_TYPE**
  Type of the SCS method.
  TYPE: String
  DEFAULT: SCS
  OPTIONS: SCS, SCSN, SCSVDW, SCSMI

- **SOS_TYPE**
  Type of the SOS method.
  TYPE: String
  DEFAULT: SOS
  OPTIONS: SOS, SOSPI

- **TRIPLES_IABC_TYPE**
  The algorithm to handle (ia|bc) type integrals that are used for (T) correction.
  TYPE: String
  DEFAULT: DISK
  OPTIONS: DISK, INCORE, AUTO, DIRECT

- **LOW_MEM_MP2**
  Do low mem mp2.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **EOM**
  Do compute eom intermediates for EOM-CCSD computations?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **EOM_MP2**
  Do compute eom_mp2 intermediates for EOM-MP2 computations?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **EKT_NEGATIVE_OCC**
  The algorithm to eliminate negative occupation numbers for density matrix in EKT computations.
  TYPE: String
  DEFAULT: ABS
  OPTIONS: ABS, SPN

  - ABS: Takes absolute values of the negative occupation numbers.
  - SPN: Replace the negative occupation numbers with a smallest positive number, such as 1.0E-15.

12 LPNOCC : Local Pair-Natural Orbital Coupled-Cluster Methods

Code Authors: U. Bozkaya and A. K. Uyar.

12.1 Overview

The LPNOCC module provides local pair-natural orbital coupled-cluster methods for reduced-scaling correlation calculations. In the current implementation, the available methods are `lpno-mp2`, `lpno-lccd`, `lpno-ccd`, `lpno-ccsd`, `lpno-ccsd(t)` and `lpno-ccsd(t1)`. Density fitting is used throughout the correlation step. The triples variants require an RHF reference; `lpno-mp2` and `lpno-ccsd` accept either, and `lpno-lccd` and `lpno-ccd` require an unrestricted one (Sec. 12.6).

Following the usage of the LPNO literature, *internal* orbitals are the occupied ones and *external* orbitals the virtuals, so that a "four-external" term is one carrying four virtual indices.

The PNO space is controlled through pair-energy and occupation-based truncation parameters. In addition, the coupled-cluster iterations may be tuned through separate convergence and DIIS options. The most important input keywords for LPNOCC are collected in the `$set_pno` and `$set_lpnocc` blocks.

12.2 Accuracy of the LPNO Approximation

When all truncation parameters are switched off (`pair_e_tol`, `eigenvalue_cutoff` and `naf_cutoff` set to zero), `lpno-ccsd` reproduces the corresponding canonical DF-CCSD correlation energy to 2×10^{-11} E_h, with no approximation left in the equations. The energy is also invariant with respect to rotations among the occupied orbitals, i.e. the localized and canonical internal orbitals give the same untruncated result.

This includes the singles dressing of the four-external ladder term,

$$R_{ij}^{\bar{a}\bar{b}} \leftarrow -\sum_{k} \sum_{\bar{c}\bar{d}} \left[(k\bar{d}|\bar{a}\bar{c}) t_k^{\bar{b}} + (k\bar{c}|\bar{b}\bar{d}) t_k^{\bar{a}} \right] \tau_{ij}^{\bar{c}\bar{d}},$$

which Neese *et al.* [J. Chem. Phys. **131**, 064103 (2009)] omit because it needs three-external integrals with an unrestricted occupied index, which their algorithm does not have. Writing it over the density-fitting factors, $(k\bar{d}|\bar{a}\bar{c}) = \sum_Q M_{ij}(k, Q, \bar{d}) B_{ij}(\bar{a}, Q, \bar{c})$, removes the obstacle: M_{ij} already runs over every occupied index, and the first of the two contractions reuses the intermediate that the ladder term itself forms. The term is worth 2.2×10^{-5} E_h for H_2O/cc-pVDZ and roughly doubles the cost of the ladder step, which is a few per cent of the iteration.

Two perturbative triples corrections are available. `lpno-ccsd(t)` uses the semicanonical (T0) approximation: the off-diagonal occupied-occupied Fock matrix elements are neglected,

which decouples the triples so that $T_{ijk}^{abc} = W_{ijk}^{abc}/D_{ijk}^{abc}$. With localized internal orbitals this introduces an error relative to canonical CCSD(T) of about 1.2×10^{-4} E_h for H_2O/cc-pVDZ.

`lpno-ccsd(t1)` instead solves the triples amplitudes iteratively, following Guo *et al.* [J. Chem. Phys. **148**, 011101 (2018)]:

$$T_{ijk}^{\bar{a}\bar{b}\bar{c}} = \frac{1}{D_{ijk}^{\bar{a}\bar{b}\bar{c}}} \left[W_{ijk}^{\bar{a}\bar{b}\bar{c}} - \sum_{l \neq k} f_{lk} \left(S T_{ijl} S^T \right)^{\bar{a}\bar{b}\bar{c}} - \sum_{l \neq j} f_{lj} \left(S T_{ilk} S^T \right)^{\bar{a}\bar{b}\bar{c}} - \sum_{l \neq i} f_{li} \left(S T_{ljk} S^T \right)^{\bar{a}\bar{b}\bar{c}} \right],$$

where S are the overlaps between the TNO spaces of two triples. This restores invariance with respect to rotations among the internal orbitals, and with the untruncated PNO and TNO spaces it reproduces canonical CCSD(T): for H_2O/cc-pVDZ the converged correction is -0.0030498722 E_h against -0.0030498719 from the canonical DF-CCSD(T) code. The iterations use a Gauss-Seidel update and converge in about four cycles. All three additional approximations of the paper are implemented. Couplings whose $|f_{lm}|$ falls below `T3_FOCK_CUTOFF` are skipped, and a triple whose own energy changes by less than a relative `T3_ITER_CUTOFF` between two sweeps is frozen for the remaining ones; the latter costs 6×10^{-9} E_h for H_2O and 1×10^{-8} E_h for n-butane while cutting the active triples from 455 to 10 over four sweeps.

The third one, running the iterations in a smaller TNO space, is available but off by default. When `T3_TNO_SCALE_STRONG` or `T3_TNO_SCALE_WEAK` is larger than one, the triples are ordered by the size of their (T0) energy, the leading ones carrying `T3_STRONG_FRACTION` of the total are called strong, and the two groups get TNO spaces truncated at the respective multiples of `EIGENVALUE_CUTOFF`. The resulting truncation is taken out again at the (T0) level, Eq. (4) of the paper. For n-butane/cc-pVDZ:

strong/weak	error in $E^{(T)}$	amplitude storage	triples step
off	–	474 MB	25.2 s
2 / 5	2.3×10^{-5} E_h	225 MB	21.2 s
3 / 10	4.1×10^{-5} E_h	160 MB	17.9 s
5 / 30	7.8×10^{-5} E_h	104 MB	14.4 s
10 / 100	1.4×10^{-4} E_h	60 MB	12.1 s

The paper's own defaults are 10 and 100, which recover more than 99.9% of the correction there. They are more aggressive here because `EIGENVALUE_CUTOFF` defaults to 10^{-7} against the 10^{-9} of the reference implementation, so the same multiple cuts much deeper. The option is therefore left off unless it is asked for; it is worth turning on when the triples amplitudes, not the time, are what limits the calculation. The triples step costs roughly three times as much as (T0), and the triples amplitudes are held in memory (the requirement is printed at the start of the iterations).

Note on naming: the paper above calls the semicanonical variant (T0) and the iterative one simply (T). The keywords here follow ORCA instead, where `(T)` is the semicanonical method and `(T1)` the iterative one. The triples variant can also be selected with the `TRIPLES_TYPE` option, which the method name overrides.

The MO Fock operator is obtained by transforming the converged AO Fock rather than by rebuilding it from the reference-basis density-fitting integrals, which is what `DF_BASIS_SCF` was needed for. For a Hartree-Fock reference the two agree to the SCF convergence – 1×10^{-9} E_h in the correlation energy at the default `D_CONVERGENCE` of 10^{-6}, and 2×10^{-11} at 10^{-10} – while the transformation costs $\mathcal{O}(N^3)$ instead of $\mathcal{O}(N^4)$; for n-hexane/cc-pVDZ it takes 8.5 s off the 12.0 s of the setup. A non-HF reference still rebuilds the operator, since there the SCF one carries the exchange-correlation potential.

The four-external integrals over PNOs are never formed: the ladder term is contracted directly against the pair-local density-fitting factors, which removes the $\mathcal{O}(n_{\mathrm{PNO}}^4)$ storage per pair and the associated disk traffic in every coupled-cluster iteration. The remaining pair-local integral blocks (the three-external $(i\bar{b}|\bar{a}\bar{c})$ and the $\langle ij|k\bar{a}\rangle$ classes) are held in memory while half of the `MEMORY` limit suffices, and re-read from disk otherwise.

The unrestricted code keeps two pair-local objects,

$$B^{ij}_{\bar{a}Q\bar{b}} \;=\; \sum_{AB} d^{ij}_{A\bar{a}}\, b^{Q}_{AB}\, d^{ij}_{B\bar{b}}, \qquad M^{ij}_{kQ\bar{a}} \;=\; \sum_{A} b^{Q}_{kA}\, d^{ij}_{A\bar{a}}, \tag{35}$$

of which only the first is stored unconditionally. M carries every internal index rather than the pair's own two, so summed over the pair list it grows as $o^3 n_Q n_{\mathrm{PNO}}$ against the $o^2 n_Q n_{\mathrm{PNO}}^2$ of B, and it overtakes B as the largest allocation in the calculation as soon as the molecule is bigger than the PNO space is wide: on the n-octane cation in cc-pVDZ it is 1.1 GB against 0.5 GB. It is also the one object that is cheap to rebuild – one matrix multiplication for a whole pair, one of $n_Q \times n_{\mathrm{PNO}} \times n_{\mathrm{vir}}$ for the single internal row that a neighbouring pair contributes. Blocks of M are therefore cached only as far as whatever is left of half the `MEMORY` limit allows once B is in memory, and rebuilt on demand beyond that. The count of resident blocks is printed before the iterations start.

The limit is thus a real control rather than a declaration: raising `MEMORY` buys speed, lowering it buys reach. With every block resident the iterations run at full speed; with none resident the same n-octane cation takes about 2.5 times longer, and that is the price of a calculation that would otherwise not start. Note that this trade is bounded – both B and M still carry the full auxiliary index, so the storage per pair is independent of how far apart the two internals are, and the pair list itself is what has to become sparse before very large molecules come into range.

12.3 PNO Module Options

- **PAIR_E_TOL**
  Pair-energy threshold used in the selection of active electron pairs.
  TYPE: Double
  DEFAULT: 1.0×10^{-4}
  OPTIONS: n User-defined value.

- **PAIR_QDEG_TOL**
  Pairs whose two internal orbital energies lie within this value of each other are kept even

when their pair energy falls below `PAIR_E_TOL`. The selection uses the semicanonical MP2 pair energy, which underestimates quasi-degenerate pairs because that is where the neglected off-diagonal occupied Fock element matters. Set to zero to select on the pair energy alone; that costs 1.2×10^{-4} E_h for the water dimer, 2.2×10^{-4} for n-butane and 3.2×10^{-4} for n-hexane in cc-pVDZ, for essentially no saving in time.
TYPE: Double
DEFAULT: 1.0×10^{-1}
OPTIONS: n User-defined value.

- **EIGENVALUE_CUTOFF**
  Occupation-number threshold used to truncate the pair-natural orbital space.
  TYPE: Double
  DEFAULT: 1.0×10^{-7}
  OPTIONS: n User-defined value.

- **NAF_CUTOFF**
  Threshold used in the natural auxiliary function (NAF) construction.
  TYPE: Double
  DEFAULT: 1.0×10^{-3}
  OPTIONS: n User-defined value.

- **NORMALIZE**
  Controls whether the generated PNOs are normalized.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **PNO_TYPE**
  Type of pair-natural orbital representation.
  TYPE: String
  DEFAULT: SEMICANONIC
  OPTIONS: SEMICANONIC, NONCANONIC

- **NORMALIZATION_TYPE**
  Controls the normalization convention used in the PNO construction. Ignored for an unrestricted reference, where the spin-orbital pair density is unambiguous.
  TYPE: String
  DEFAULT: PSI4
  OPTIONS: PSI4, ORCA

- **CC_CHECK**
  Runs a brute-force spin-orbital LCCD or CCD alongside `lpno-lccd` / `lpno-ccd` as an independent reference (Sec. 12.7). It stores the four-external integrals in full, so it is for small cases only.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **CC_AMP_TOL**
  Amplitude-norm threshold for the pair–pair terms of the unrestricted coupled cluster (Sec. 12.7). A neighbouring pair whose amplitude has a smaller Frobenius norm is skipped. Zero disables the screening.
  TYPE: Double
  DEFAULT: 0.0
  OPTIONS: n User-defined value.

- **CC_CHECK_TOL**
  Tolerance on the difference between the spin-integrated energy and the brute-force reference when `CC_CHECK` is on; exceeding it stops the calculation. Only meaningful with a complete PNO space, since the reference works in the full external space.
  TYPE: Double
  DEFAULT: 1.0×10^{-8}
  OPTIONS: n User-defined value.

- **CC_TERMS_S**
  Bitmask over the nine groups the CCSD singles split into: 16 the T1 equation, 32 F_{ae}, 512 F_{mi}, 256 the F_{me} dressings of the T2 Fock terms, 64 τ and $\tilde{\tau}$, 1024 W_{mnij}, 2048 W_{abef}, 4096 W_{mbej}, 8192 the ring term quadratic in t_1. Honoured by the brute-force reference as well. Setting it to 0 recovers `lpno-ccd`.
  TYPE: Integer
  DEFAULT: 16368 (all)
  OPTIONS: 0–16368

- **CC_TERMS**
  Bitmask over the four quadratic terms of `lpno-ccd`: 1 for F_{be}, 2 for F_{mj}, 4 for the ladder–ladder term, 8 for the ring dressing. Honoured by the brute-force reference as well, so a term can be compared on its own. Setting it to 0 recovers `lpno-lccd`.
  TYPE: Integer
  DEFAULT: 15
  OPTIONS: 0–15

- **MP2_MAXITER**
  Maximum number of iterations in the `lpno-mp2` amplitude solve (Sec. 12.5). Setting it to 1 recovers the non-iterative pair energies.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **MP2_CONVERGENCE**
  RMS amplitude-change threshold for the `lpno-mp2` amplitude solve.
  TYPE: Double
  DEFAULT: 1.0×10^{-10}
  OPTIONS: n User-defined value.

12.4 LPNOCC Module Options

- **CC_MAXITER**
  Maximum number of coupled-cluster iterations.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **E_CONVERGENCE**
  Energy convergence threshold for the coupled-cluster iterations.
  TYPE: Double
  DEFAULT: 1.0×10^{-6}
  OPTIONS: n User-defined value.

- **R_CONVERGENCE**
  Residual convergence threshold for the amplitude equations.
  TYPE: Double
  DEFAULT: 1.0×10^{-5}
  OPTIONS: n User-defined value.

- **CC_DIIS_MAX_VECS**
  Maximum number of DIIS vectors stored during the iterations.
  TYPE: Integer
  DEFAULT: 6
  OPTIONS: n User-defined value.

- **CC_DIIS_MIN_VECS**
  Minimum number of DIIS vectors required before extrapolation starts.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: n User-defined value.

- **DO_DIIS**
  Activates DIIS acceleration for the coupled-cluster iterations.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **LEVEL_SHIFT**
  Level-shift parameter used in the DIIS procedure.
  TYPE: Double
  DEFAULT: 0.02
  OPTIONS: n User-defined value.

- **TRIPLES_TYPE**
  Selects the perturbative triples variant. Overridden by the method name, i.e. `lpno-ccsd(t1)` always requests T1.

TYPE: String
DEFAULT: T0
OPTIONS: T0 (semicanonical), T1 (iterative)

- **T3_MAXITER**
  Maximum number of iterations for the T1 triples amplitudes.
  TYPE: Integer
  DEFAULT: 20
  OPTIONS: n User-defined value.

- **T3_CONVERGENCE**
  Energy convergence threshold for the T1 triples iterations.
  TYPE: Double
  DEFAULT: 1.0×10^{-8}
  OPTIONS: n User-defined value.

- **T3_FOCK_CUTOFF**
  Couplings between triples whose occupied Fock matrix element is smaller than this value are neglected.
  TYPE: Double
  DEFAULT: 1.0×10^{-3}
  OPTIONS: n User-defined value.

- **T3_ITER_CUTOFF**
  A triple is frozen for the remaining iterations once its own energy changes, relative to itself, by less than this value between two sweeps. Set to zero to iterate every triple to convergence.
  TYPE: Double
  DEFAULT: 1.0×10^{-3}
  OPTIONS: n User-defined value.

- **T3_TNO_SCALE_STRONG**, **T3_TNO_SCALE_WEAK**
  Multiples of `EIGENVALUE_CUTOFF` at which the TNO spaces of the strong and weak triples are truncated for the T1 iterations. Both default to one, which disables the reduction.
  TYPE: Double
  DEFAULT: 1.0
  OPTIONS: n User-defined value.

- **T3_STRONG_FRACTION**
  Fraction of the total (T0) energy that the strong triples must carry.
  TYPE: Double
  DEFAULT: 0.9
  OPTIONS: n User-defined value.

12.5 LPNO-MP2 and the Internal Fock Block

The pair energies that define the PNOs are built from $t = K/D$, which assumes the internal block of the Fock matrix is diagonal. It is not. The whole point of a local method is to run on localized internal orbitals, and localizing them fills in f_{ik}. Dropping those elements costs far more than the PNO truncation ever does and, worse, destroys invariance with respect to rotations among the internal orbitals — the property that makes a local method a method rather than a recipe.

`lpno-mp2` therefore solves the first-order equations. In spin orbitals,

$$0 = K_{ab}^{ij} + \sum_c f_{ac} t_{cb}^{ij} + \sum_c f_{bc} t_{ac}^{ij} - \sum_k f_{ki} t_{ab}^{kj} - \sum_k f_{kj} t_{ab}^{ik}.$$

The external sums are trivial, the PNOs of a pair being semicanonical. The internal sums are the work, because t^{kj} lives in the PNO space of pair kj and has to be carried into that of pair ij,

$$[t^{kj}]^{ij} = S^{ij,kj}\, t^{kj}\, (S^{ij,kj})^T, \qquad S^{ij,kj} = (d^{ij})^T d^{kj}.$$

Separating $k = i$ and $k = j$ from those sums gives the denominator and leaves a Jacobi iteration. The three spin blocks of an unrestricted calculation do not mix, the Fock operator being spin diagonal.

For a reference that does not satisfy Brillouin's theorem — quasi-restricted orbitals (Sec. 8.3) being the case that matters — f_{ia} is nonzero and carries first-order singles, $0 = f_{ia} + \sum_c f_{ac} t_i^c - \sum_k f_{ki} t_k^a$, contributing $\sum_{ia} f_{ia} t_i^a$ to the energy. These are kept in the full external space; there are only $O(N^2)$ of them.

The effect is large and the cost is not. With localized internals and nothing truncated, the energy now comes out at the canonical DF-MP2 value exactly — to all fourteen printed digits, for water in cc-pVDZ ($-0.20179181534679\ E_h$) and for the methyl radical through the unrestricted path ($-0.12652154006442\ E_h$) — where the non-iterative pair energies give -0.19532223 and -0.12498805 respectively. At production thresholds, for n-butane in cc-pVDZ:

	correlation energy	error vs canonical
non-iterative pair energies	$-0.5755026\ E_h$	$1.5 \times 10^{-2}\ E_h$
`lpno-mp2`	$-0.5903003\ E_h$	$5.6 \times 10^{-4}\ E_h$

a factor of 27 in accuracy for 13 iterations and 0.04 s, against 1.2 s for the calculation as a whole. With canonical internals the solver converges on the first pass and changes nothing.

A QRO reference gives the same energy whether or not the internals are localized, singles included, which is the sharpest available check that the solver is invariant in the way it claims.

12.6 Open-Shell LPNO-MP2

`lpno-mp2` accepts an unrestricted reference. The pair list then splits into three spin cases, which are all treated in spin orbitals: for a pair of internal spin orbitals (i, j) the first-order

amplitudes are

$$t_{ij}^{ab} = \frac{\langle ij||ab\rangle}{f_{ii} + f_{jj} - f_{aa} - f_{bb}},$$

and the pair density that defines the PNOs is

$$D_{ab}^{ij} = \sum_{c} \left(t_{ac}^{ij} t_{bc}^{ij} + t_{ca}^{ij} t_{cb}^{ij} \right).$$

No $(1+\delta_{ij})$ normalization appears: a same-spin pair is strictly $i > j$ and a mixed-spin pair is counted once, so no pair is ever diagonal, and the `normalization_type` option is therefore ignored for an unrestricted reference.

Splitting the two terms by spin case gives the structure of the implementation. For a same-spin pair both external labels carry the same spin and t^{ij} is antisymmetric, so the two terms coincide and the pair carries one PNO set. For a mixed-spin pair $(I\alpha, j\beta)$ the amplitude t_{Ab}^{Ij} is rectangular and the two terms are not two contributions to one matrix — they are the α block and the β block,

$$D^{Ij}(\alpha)_{AB} = \sum_{b} t_{Ab}^{Ij} t_{Bb}^{Ij}, \qquad D^{Ij}(\beta)_{ab} = \sum_{B} t_{Ba}^{Ij} t_{Bb}^{Ij},$$

so a mixed pair carries *two* PNO sets, one per spin and of independent sizes, and everything living on such a pair is rectangular in the PNO basis.

The truncation threshold means much the same thing as in the closed-shell code. Summing the four spin pairs that one closed-shell spatial pair (i, j) breaks into gives, for the α externals, $3\,\mathbf{t}\mathbf{t}^T + 3\,\mathbf{t}^T\mathbf{t} - 2\,\mathbf{t}\mathbf{t} - 2\,\mathbf{t}^T\mathbf{t}^T$, against the spin-summed closed-shell density $4\,\mathbf{t}\mathbf{t}^T + 4\,\mathbf{t}^T\mathbf{t} - 2\,\mathbf{t}\mathbf{t} - 2\,\mathbf{t}^T\mathbf{t}^T$. The two differ, as they must, but only by a factor of order one.

With every pair kept, every PNO kept and the auxiliary basis uncompressed, the method reproduces canonical UHF DF-MP2 exactly. For the methyl radical in cc-pVDZ all fourteen printed digits of each spin component agree with the DFOCC module,

	DFOCC DF-MP2	untruncated LPNO-MP2
$\alpha\alpha$	−0.01817011607314	−0.01817011607314
$\beta\beta$	−0.00634408308892	−0.00634408308892
$\alpha\beta$	−0.10200734090236	−0.10200734090236
total	−0.12652154006442	−0.12652154006442

and tightening `eigenvalue_cutoff` from 10^{-5} to 10^{-10} takes the error from 1.4×10^{-3} to below 10^{-14} E_h monotonically. A closed-shell molecule run through the unrestricted path agrees with the restricted one to 2×10^{-13} E_h. Both the untruncated and the production-threshold cases are covered by the `lpnocc` test set.

Two numbers are printed for an unrestricted run: the energy from the pair amplitudes, which uses the diagonal of the external Fock block, and the energy rebuilt inside the truncated PNO spaces, which uses its eigenvalues within each pair space. For a canonical reference these agree and their difference is purely the truncation error. For a non-canonical one they do not,

and the module says so rather than let the difference be misread as a truncation measure. Quasi-restricted orbitals (Sec. 8.3) are the case that matters here: their SOMO and VMO blocks are canonicalized against different operators, so the external Fock block retains large off-diagonal elements. QROs may be combined with `lpno-mp2`, but note that Brillouin's theorem does not hold for them, so the singles contribution their reference carries is not yet included.

Open-shell coupled cluster is not available; `lpno-ccsd` and its triples variants stop with an error for an unrestricted reference.

12.7 Open-Shell LPNO-LCCD and LPNO-CCD

`lpno-lccd` solves the linearized doubles equations over an unrestricted pair list. In spin orbitals,

$$R_{ab}^{ij} = \langle ij||ab\rangle + P(ab)\sum_e f_{be}t_{ae}^{ij} - P(ij)\sum_m f_{mj}t_{ab}^{im} + \tfrac{1}{2}\sum_{ef}\langle ab||ef\rangle t_{ef}^{ij} + \tfrac{1}{2}\sum_{mn}\langle mn||ij\rangle t_{ab}^{mn} + P(ij)P(ab)\sum_{me}\langle mb||ej\rangle t_{ae}^{im}.$$

Spin-integrating this is where the work lies, and one rule keeps it tractable: carry every amplitude into the PNO space of the pair whose residual is being built, after which every integral is that pair's own. The step is exact when nothing is truncated, the overlap being unitary then, and it is the LPNO approximation itself when something is.

Two consequences are worth stating. A same-spin residual draws on mixed-spin amplitudes and vice versa — the ring term couples all three blocks, so they cannot be converged separately. And on a mixed pair only the α index of an $\alpha\beta$ amplitude can be carried into the α space; the β index has to be met by an integral built from the *other* pair's β PNOs, which is why the three-index intermediate M carries every internal index rather than just the pair's own two.

Validation

Three independent checks agree. Untruncated, the correlation energy reproduces canonical UHF DF-LCCD: -0.15255115125800 against -0.15255115125981 for the methyl radical, -0.35177553947715 against -0.35177553947886 for triplet O_2, and -0.21380803422843 against -0.21380803422896 for water pushed through the unrestricted path — all cc-pVDZ, all agreeing to about 10^{-12} E_h, and each spin component separately. (The dfocc references have to have their own convergence tightened; at the module defaults they are converged only to 10^{-8} and cannot serve as references at this level.)

Second, `$set_pno cc_check true` runs a brute-force spin-orbital LCCD alongside — one index range covering both spins, one antisymmetrized integral array, the expression above written out literally, $\langle ab||cd\rangle$ stored in full so it is only usable on small cases. It agrees with the spin-integrated code to all fourteen printed digits, block by block. A nearly-right energy

is worse than an obviously wrong one, because it looks like convergence; this is the check that distinguishes them.

Third, with localized internals the truncation error falls monotonically towards zero as `eigenvalue_cutoff` is tightened.

The quadratic terms

`lpno-ccd` adds the four T_2^2 terms,

$$F_{be} = -\tfrac{1}{2}\sum_{mnf} t_{bf}^{mn}\langle mn||ef\rangle, \qquad F_{mj} = +\tfrac{1}{2}\sum_{nef} t_{ef}^{jn}\langle mn||ef\rangle,$$

$$W_{mnij} \mathrel{+}= \tfrac{1}{4}\sum_{ef} t_{ef}^{ij}\langle mn||ef\rangle, \qquad W_{abef} \mathrel{+}= \tfrac{1}{4}\sum_{mn} t_{ab}^{mn}\langle mn||ef\rangle,$$

$$W_{mbej} \mathrel{+}= -\tfrac{1}{2}\sum_{nf} t_{fb}^{jn}\langle mn||ef\rangle.$$

The two written on the middle line are the same contraction once expanded — $\frac{1}{8}\sum_{mnef}\langle mn||ef\rangle t_{ef}^{ij} t_{ab}^{mn}$ each — and together they make the familiar $\frac{1}{4}$ of the ladder–ladder term. Implementing only the W_{mnij} half is a 5.5×10^{-4} E_h error that converges cleanly and looks like an answer.

The first two need no new spin integration: they dress f_{be} and f_{mj}, which the linear terms already handle. The only care needed is the diagonal, which the linear code hides in the denominator and the dressing does not. The ring dressing is the laborious one, surviving in six spin cases, each picking up a different pair of contributions; two of them — those where m and j share a spin and b and e share the other — keep only the exchange half of $\langle mn||ef\rangle$ and so have a single term.

Each of the four is validated separately against the brute-force solver through `$set_pno cc_terms`, a bitmask over the four (1, 2, 4, 8; default 15). Comparing only totals tells you that something is wrong and nothing about what; comparing term by term found a sign in F_{mj} and a factor of two in the ladder–ladder term that no amount of rereading had. Untruncated, all four agree to fourteen digits, and the totals reproduce canonical UHF DF-CCD: -0.14878175717069 against -0.14878175717211 for the methyl radical, -0.34353084537908 against -0.34353084537912 for triplet O_2, and -0.21077195215645 against -0.21077195215693 for water.

Convergence

The unrestricted amplitude equations are extrapolated with DIIS. The amplitudes live in per-pair spaces of different sizes, so the extrapolation works on one flat vector holding every pair's block back to back plus the two singles blocks, rather than a separate DIIS per pair: the pairs are coupled, every residual drawing on its neighbours' amplitudes, so extrapolating each independently would not produce a consistent amplitude set. This is the same arrangement

DFOCC uses for its coupled cluster, which passes singles and doubles as two data blocks of one manager.

For the n-butane cation in cc-pVDZ at default thresholds this takes the calculation from 33 iterations and 253 s to 12 iterations and 80 s. It also changes the answer: without extrapolation the energy is still drifting when the default ΔE criterion trips, and the run stops 1.8×10^{-6} E_h short. Controlled by `DO_DIIS`, `CC_DIIS_MIN_VECS` and `CC_DIIS_MAX_VECS` in the `$set_lpnocc` block.

Cost

Three things decide how the unrestricted coupled cluster scales, and they are worth knowing because two of them are settled in the code and one is left to the user.

F_{be} has free external indices, so the obvious implementation rebuilds it in the space of every target pair — which puts a loop over all pairs inside the loop over pairs. It does not have to: F is one matrix over the canonical external space, each pair contributes to it in its own PNO basis and is back-transformed once, and each target pair then projects the finished matrix in. That is $O(N_{\mathrm{pair}})$ instead of $O(N_{\mathrm{pair}}^2)$, and it changes the answer only in how the truncation approximation is made, the shared build carrying one projection where the per-pair build carried two.

Most of the ring dressing comes apart in the same way. Written out, three of its four pieces contract the neighbouring amplitude with the auxiliary index before ever meeting the summed internal, so they collapse into two objects built once per pair; only the exchange piece, where both internals sit inside the auxiliary sum, stays a double loop.

What is left — the internal ladder, the ladder–ladder term and that exchange piece — is a genuine pair–pair coupling, and what saves it is that a pair whose internals are far apart carries a vanishing amplitude. `CC_AMP_TOL` skips a neighbour whose amplitude norm falls below it, which is what turns those loops from $O(N_{\mathrm{pair}})$ to $O(1)$ per pair for an extended system. The norm of the stored amplitude bounds the norm of its projection, so nothing has to be projected to decide. It is off by default, and on the small test cases here it buys little, because the pair prescreening of `PAIR_E_TOL` has already removed the distant pairs before it is reached; for a separated $CH_3 \cdots H_2O$ pair at 12 Å it costs 8×10^{-9} E_h at 10^{-4}.

For the n-butane cation in cc-pVDZ on eight threads, four iterations, 300 surviving pairs and 42–51 PNOs each:

	before	after
`lpno-lccd`	22.7 s	6.6 s
`lpno-ccd`	135.0 s	22.0 s
PNO integrals	1.2 s	0.4 s

The largest single factor was not algorithmic: the residual assembly was written with explicit index loops, and replacing them with the tensor library's BLAS calls was worth more than everything else together. Every contraction in the module now goes through `contract`, `gmm`, `gtrans`, `axpy` or `dot`.

Choosing the threshold

A given `eigenvalue_cutoff` buys a leaner space here than in the closed-shell code, and the reason is structural rather than numerical. The α density of a mixed pair is $\mathbf{t}\mathbf{t}^T$ alone, where the closed-shell spatial pair — which lumps the spin pairs $(i\alpha j\beta)$ and $(i\beta j\alpha)$ together — combines $\mathbf{t}\mathbf{t}^T$ and $\mathbf{t}^T\mathbf{t}$ and is several times larger in trace. Same-spin densities are leaner again, being built from the intrinsically weak same-spin amplitudes. For the methyl radical in cc-pVDZ with localized internals:

`eigenvalue_cutoff`	PNOs per pair (AA/BB/AB)	error
10^{-7}	15.0 / 18.0 / 16.8	2.2×10^{-3} E_h
10^{-10}	15.0 / 18.0 / 21.3	1.6×10^{-3} E_h
10^{-14}	20.0 / 18.0 / 23.6	2.3×10^{-4} E_h
10^{-18}	22.7 / 23.3 / 24.0	1.1×10^{-4} E_h

so the closed-shell default of 10^{-7} is too loose for an unrestricted calculation; treat 10^{-10} as a starting point and tighten from there.

One trap is worth knowing. With canonical orbitals in a symmetric molecule the pair densities are genuinely rank deficient — symmetry forbids most of the amplitude blocks — and their spectra fall off a cliff, for the methyl radical from 7×10^{-8} straight to 10^{-20}. No positive threshold then reaches the exact answer, because "keep every PNO with a nonzero occupation" is not the same thing as "span the external space". Setting `eigenvalue_cutoff` negative keeps the numerically null directions as well and does recover it; that is what the untruncated test in the `lpnocc` set does. Localizing the internals breaks the symmetry and removes the cliff.

12.8 Open-Shell LPNO-CCSD(T)

`METHOD LPNO-CCSD(T)` with an unrestricted reference adds the semicanonical perturbative triples. In spin orbitals there is a single equation,

$$W_{ijk}^{abc} = P(i/jk)\,P(a/bc)\Big[\sum_e t_{ae}^{jk}\,\langle ei\|bc\rangle - \sum_m t_{bc}^{im}\,\langle ma\|jk\rangle\Big], \tag{36}$$

$$V_{ijk}^{abc} = P(i/jk)\,P(a/bc)\Big[t_i^a\,\langle jk\|bc\rangle + t_{bc}^{jk}\,f_{ia}\Big], \tag{37}$$

$$E^{(T)} = \tfrac{1}{36}\sum_{ijkabc}\frac{W_{ijk}^{abc}\left(W_{ijk}^{abc}+V_{ijk}^{abc}\right)}{D_{ijk}^{abc}}, \tag{38}$$

but spin integration splits it into four cases — $\alpha\alpha\alpha$, $\alpha\alpha\beta$, $\alpha\beta\beta$, $\beta\beta\beta$ — which share no algebra and are therefore written and validated separately. The same-spin cases carry six terms and the mixed ones ten. Each is reported on its own line of the output, and the brute-force reference solver reached through `PNO/CC_CHECK` reports the same four numbers, so a disagreement points at one case rather than at a total.

The f_{ia} term of the disconnected part is not optional. It vanishes for a canonical Hartree-Fock determinant, which is why most write-ups drop it, but the open-shell code is meant to run on quasi-restricted orbitals, where Brillouin's theorem does not hold.

Triple natural orbitals

Each triple gets its own external space, obtained by diagonalizing the average of the three pair densities it stands on. Those densities are never rebuilt in the full external space: a pair stores its PNO coefficients together with the matching occupation numbers, and $\mathbf{d}\,\mathbf{n}\,\mathbf{d}^T$ is the truncated pair density exactly. A mixed triple carries *two* TNO sets of independent sizes, for the same reason a mixed pair carries two PNO sets — $\alpha\alpha\beta$ averages three pair densities for its α space and two for its β one. The truncation uses `EIGENVALUE_CUTOFF`, as in the closed-shell triples.

A triple is visited exactly once, so its natural orbitals are built where they are used and discarded afterwards. Storing them would cost $\mathcal{O}(o^3vn)$ for no benefit; the closed-shell code keeps them only because its iterative `(T1)` variant revisits them.

The external space is semicanonicalized before the energy expression, which is what makes the diagonal denominator D_{ijk}^{abc} exact in that space rather than an approximation to it. This matters on quasi-restricted orbitals: the virtuals are canonicalized against $F(\alpha)$, so $F(\beta)$ retains an off-diagonal external block, and neglecting it instead of rotating it away moves the methyl radical's $E^{(T)}$ by $8 \times 10^{-6}\ E_h$.

Validation

For the methyl radical in cc-pVDZ with nothing truncated, the four spin cases agree with the brute-force spin-orbital solver and with canonical UHF DF-CCSD(T) from DFOCC to every digit printed:

	canonical UHF	QRO
$\alpha\alpha\alpha$	−0.00006102569211	−0.00004751521981
$\alpha\alpha\beta$	−0.00162994960298	−0.00159905187688
$\alpha\beta\beta$	−0.00089041608649	−0.00090736048727
$\beta\beta\beta$	−0.00000882855646	−0.00001000839050
$E^{(T)}$	−0.00259021993801	−0.00256393597446

The quasi-restricted column found a defect that no canonical test could: the singles live in the full external space, so they need the off-diagonal external Fock block explicitly, and it was missing. With a canonical reference that block is zero and the omission is invisible; on QROs it was worth $2 \times 10^{-4}\ E_h$ in the CCSD energy alone. Both `CC_CHECK` tests now run in each reference.

12.9 Behaviour with Truncation

At the default thresholds the truncation is real and its size is worth knowing. For the water dimer in cc-pVDZ, 5 of the 36 pairs are dropped and the PNO spaces average 17.5 of the 38 virtuals, which gives

	correlation energy	error vs canonical
`lpno-ccsd`	$-0.4245039\ E_h$	$2.1 \times 10^{-4}\ E_h$
`lpno-ccsd(t)`	$-0.0061001\ E_h$ (triples)	$5.0 \times 10^{-4}\ E_h$
`lpno-ccsd(t1)`	$-0.0063418\ E_h$ (triples)	$2.6 \times 10^{-4}\ E_h$

so 99.95% of the CCSD correlation energy is recovered, and the iterative triples halve the error of the semicanonical ones. Both are covered by the `lpnocc` test set.

12.10 Input Examples

*Single-point `lpno-ccsd/cc-pvdz` computation:

```
$set_molecule H2O
  0 1
  O   0.00000   0.00000  -0.06577
  H   0.00000  -0.75906   0.52195
  H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method lpno-ccsd
  reference rhf
  jobtype energy
  freeze_core true
$end_globals

$set_pno
  pair_e_tol 1.0e-4
  eigenvalue_cutoff 1.0e-7
  naf_cutoff 1.0e-3
  pno_type semicanonic
  normalization_type psi4
$end_pno

$set_lpnocc
  e_convergence 1.0e-6
```

```
  r_convergence 1.0e-5
  cc_maxiter 50
  do_diis true
$end_lpnocc
```

*Single-point `lpno-ccsd(t)/cc-pvdz` computation:

```
$set_molecule H2O
  0 1
  O   0.00000   0.00000  -0.06577
  H   0.00000  -0.75906   0.52195
  H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method lpno-ccsd(t)
  reference rhf
  jobtype energy
  freeze_core true
$end_globals

$set_pno
  pair_e_tol 1.0e-4
  eigenvalue_cutoff 1.0e-7
  naf_cutoff 1.0e-3
$end_pno

$set_lpnocc
  e_convergence 1.0e-6
  r_convergence 1.0e-5
  cc_maxiter 50
$end_lpnocc
```

13 CIM: Cluster-in-Molecule Methods

Code Authors: Y. Alagöz.

13.1 Overview

The Cim module provides cluster-in-molecule (CIM) local-correlation methods, a reduced-scaling route to post-Hartree–Fock correlation energies that was introduced by Li and cowork-

ers [Li, Ma, and Jiang, J. Comput. Chem. **23**, 237 (2001); Li, Shen, Li, and Jiang, J. Chem. Phys. **125**, 074109 (2006)]. Rather than truncating the virtual space pair by pair, as the LPNOCC and DLPNOCC modules do (Secs. 12–**??**), CIM splits the correlated occupied space itself into overlapping, chemically local clusters, or *subsystems*, runs an ordinary correlated calculation on each one in its own reduced orbital and AO basis, and reassembles the total correlation energy from the orbital-energy contributions of the subsystems' *central* orbitals. The available methods are `cim-mp2`, `cim-mp2.5`, `cim-mp3`, `cim-ccsd`, and `cim-ccsd(t)`; the spin-component-scaled MP2 and MP3 energies are reported alongside whenever `DO_SCS` is on in the `$set_dfocc` block.

Two schemes are available for building the subsystems, selected with `CIM_TYPE`: single-environment CIM (SECIM, the default) and improved CIM (ICIM). SECIM follows the design of Li and Piecuch [J. Phys. Chem. A **114**, 8644 (2010)], itself a simplification of the two-threshold dual-environment CIM (DECIM) scheme of Li, Piecuch, Gour, and Li [J. Chem. Phys. **131**, 114109 (2009)]. ICIM follows Guo, Li, and Li [J. Phys. Chem. A **118**, 8996 (2014)]. The two differ in how a subsystem's occupied and virtual orbitals are built (Secs. 13.3 and 13.4), but share the same underlying energy decomposition (Sec. 13.2). SECIM requires `FREEZE_CORE TRUE`; ICIM places no such restriction on the reference.

13.2 Localization and the CIM Energy Decomposition

The core and active occupied orbitals of the whole-molecule Hartree–Fock reference are localized separately with the Boys criterion [Foster and Boys, Rev. Mod. Phys. **32**, 300 (1960)], giving the localized molecular orbitals (LMOs) ϕ_i that everything else in this module is built on. The correlation energy of a size-consistent correlated method, written in terms of these LMOs, is a sum of per-orbital contributions,

$$E_{\mathrm{corr}} = \sum_i \delta E_i \,,$$

where, for CCSD,

$$\delta E_i = \sum_a t_i^a f_{ia} + \frac{1}{4} \sum_{j,a,b} \tau_{ij}^{ab} \langle ij \| ab \rangle \,,$$

with the sums over a, j, b running over the full virtual and occupied spaces of the molecule. This identity is exact and holds for any choice of occupied localization, but it is also the starting point for the approximation: if δE_i is dominated by amplitudes and integrals that involve orbitals close to ϕ_i, replacing the full occupied and virtual spaces by a smaller *domain* built around ϕ_i should reproduce δE_i to good accuracy at a fraction of the cost, since the domain, and not the whole molecule, sets the size of the post-HF calculation.

Every LMO in the domain that was the reason the domain was built – as opposed to one dragged in only to polarize the domain's virtual space – is called *central* in that domain; the module never lets an active occupied LMO be central in more than one surviving domain (Secs. 13.3 and 13.4 describe how domains that would leave an LMO central in several are merged), so the sum over i above is realized, subsystem by subsystem, as a sum over each subsystem's central LMOs alone. The `PREFACTOR` attached to every LMO – the reciprocal

of the number of surviving domains in which it is central – exists precisely to keep this true even when `COLLAPSE` is turned off and a genuinely redundant set of domains is used instead (Sec. 13.3).

13.3 Single-Environment CIM (SECIM)

SECIM builds one domain per non-hydrogen atom (together with the hydrogens bonded to it, a bond being recognized when the interatomic distance is below the sum of the van der Waals radii of the two atoms plus 0.168 Å), rather than one per LMO, which keeps the number of domains comparable to the number of heavy atoms instead of the number of occupied orbitals.

13.3.1 Central and Environmental LMOs

For every LMO ϕ_i, a Mulliken population analysis gives its charge $q_A^{(i)}$ on every atom A,

$$q_A^{(i)} = \sum_{\mu \in A} \left(\boldsymbol{D}^{(i)} \boldsymbol{S} \right)_{\mu\mu} , \qquad D_{\mu\nu}^{(i)} = C_{\mu i} C_{\nu i} ,$$

where $\boldsymbol{S}$ is the AO overlap matrix. An LMO is assigned as central to a non-hydrogen atom (or to one of its bonded hydrogens) whenever its Mulliken charge there is at least `ORB_TOL`. Once every atom has its set of central LMOs, the environmental LMOs of that atom's domain are picked out through the Fock matrix: ϕ_j is added to the domain of a central LMO ϕ_i whenever $|f_{ij}| \geq$ `ENV_TOL` and ϕ_j is not itself central there. A single threshold does the work that DECIM splits into a primary and a secondary environment threshold, which is the simplification the S in SECIM refers to.

13.3.2 Collapsing Embedded Domains

Domains built this way overlap heavily, and a domain that is wholly contained in a larger one contributes nothing that the larger one does not already provide. With `COLLAPSE TRUE` (the default), such embedded domains are merged into the larger one, their central LMOs becoming additional central LMOs of the survivor, which is what keeps every active LMO central in exactly one domain and lets `PREFACTOR` be trivially 1 everywhere. With `COLLAPSE FALSE` the raw, redundant set of domains is kept instead; an LMO that is central in several of them then has its contribution to E_{corr} split, via `PREFACTOR`, across every domain that computes it.

13.3.3 AO Domains and Buffer Atoms

Each LMO also gets an AO domain: the atoms whose Mulliken charge for that LMO is at least `ORB_TOL`, the same threshold used to pick central LMOs above. The AO domain of a subsystem is the union of its member LMOs' AO domains, and every atom of the molecule

must appear in at least one LMO's AO domain – if `ORB_TOL` is set too high for some atom to ever pass the threshold, the module stops and asks for a smaller value. The *extended* subsystem then adds a buffer: every non-hydrogen atom within `BUFF_DISTANCE` (Å) of any subsystem atom, plus the hydrogens bonded to the subsystem and its buffer, together with the core LMOs and any additional active occupied LMOs whose own AO domain lies entirely inside the extended set.

13.3.4 Projected Orbitals and Virtual Space

The LMOs assigned to the extended subsystem are projected, Boughton–Pulay style [Boughton and Pulay, J. Comput. Chem. **14**, 736 (1993)], onto the extended AO domain and renormalized; projected atomic orbitals (PAOs) are then built from the orthogonal complement of these projected occupied orbitals and canonically orthogonalized, discarding directions whose overlap eigenvalue falls below `ORTH_CUTOFF`, which gives the extended subsystem's virtual space. Both the occupied and the virtual orbitals of the extended subsystem are then projected a second time, down onto the AO domain of the plain (non-extended) subsystem. Every projected occupied orbital is kept, but a projected virtual is retained only if the squared overlap between the once- and twice-projected orbital is at least 1− `VIR_CUTOFF`; if none pass, the run stops and asks for a larger `VIR_CUTOFF`. The two projection steps – extended subsystem, then subsystem – are what let a domain built from a handful of central and environmental LMOs end up with an occupied and virtual space confined to a correspondingly small set of atoms.

13.4 Improved CIM (ICIM)

ICIM assigns exactly one *central atom* to every LMO – the atom of largest absolute Mulliken charge, using the same $q_A^{(i)}$ as above – rather than grouping several LMOs under one heavy atom the way SECIM does, and it measures orbital proximity by the distance between central atoms instead of by the Fock matrix. The MO domain of an LMO ϕ_i consists of ϕ_i itself, as the sole central orbital, together with every LMO whose central atom lies within `ICIM_TOL` (Å) of ϕ_i's own central atom, as environmental orbitals; embedded domains are merged exactly as in SECIM when `COLLAPSE` is on.

13.4.1 AO Domain of an MO Domain

Starting from the (unique) central atoms of an MO domain's member LMOs, the AO domain is grown radially, atom shell by atom shell, until every member LMO can be projected onto it with a least-squares error below a fixed internal tolerance; this is the `atoms_of_subsyss` domain and it plays the same role SECIM's AO domain plays, but it is built from the MO domain directly rather than from a union of individually thresholded per-LMO domains.

13.4.2 Approximate PAOs and the Composite Domain

PAOs for the entire molecule are built once, from the orthogonal complement of every active LMO's density, following Pulay and Saebø [Pulay, Chem. Phys. Lett. **100**, 151 (1983); Saebø and Pulay, Chem. Phys. Lett. **113**, 13 (1985)]. Reproducing every one of these molecule-spanning PAOs exactly for each subsystem would defeat the purpose of localizing the calculation, so ICIM instead builds, once per atom A and independently of any particular subsystem, an *approximate* PAO (APAO) domain: starting from A and its neighbours within 2.0 Å, the domain is grown until the PAOs centered on A can be reproduced, in the same least-squares sense as above, to within `APAO_CUTOFF`. The composite AO domain of a given subsystem is then simply the union of the precomputed APAO domains of the central atoms of every LMO – central or environmental – in its MO domain. This atom-by-atom precomputation is what makes the composite domain cheap to assemble for every subsystem in turn, and it is also the reason ICIM needs no analogue of SECIM's separate extended-subsystem step.

13.4.3 Virtual Space

The occupied orbitals of a subsystem are its LMOs projected onto the composite AO domain, exactly as in SECIM. The virtual space, however, is obtained in one step rather than two: the PAOs of the (non-composite) subsystem AO domain are projected into the composite domain and canonically orthogonalized there, again discarding directions below `ORTH_CUTOFF`; there is no separate retention threshold analogous to SECIM's `VIR_CUTOFF`; the canonical orthogonalization by itself removes the near-redundant combinations that a second projection step would otherwise be needed to filter out.

13.5 Semicanonicalization and Assembling the Total Energy

Before the correlated calculation, each subsystem's occupied-occupied and virtual-virtual blocks of the Fock matrix, evaluated in that subsystem's own projected orbitals, are diagonalized separately, and the subsystem is run in the resulting quasi-canonical basis; this semicanonicalization is not an optional refinement in `MacroQC`, it is always applied, since the projected orbitals of a domain do not themselves diagonalize the Fock matrix and the coupled-cluster iterations converge far better in the basis that does. Each subsystem is otherwise an ordinary density-fitted DFOCC calculation, run in the subsystem's own reduced molecule, AO basis, and orbital space; the frame translation applied to the original geometry is carried over to every subsystem unchanged and reorientation is left off, so a subsystem's atoms coincide exactly with the corresponding atoms of the parent molecule rather than drifting to their own center of mass. Only the subsystem's central LMOs contribute to the total: their per-orbital energies, weighted by `PREFACTOR` (Sec. 13.2), are summed over every domain to give E_{corr}. `DRY_RUN TRUE` still builds every subsystem's projected orbitals, Fock matrix, and density-fitting integrals and prints the domain-size summary table, but skips the DFOCC correlation step itself, which is normally what the wall time is spent on; this is a quick way to see how large the subsystems have become, and how many of them

there are, before committing to a full run. `WRITE_LOCAL_MOLDEN TRUE` writes the whole-molecule Boys-localized orbitals, before any projection has taken place, to a Molden file; `WRITE_SUBSYS_MOLDEN TRUE` writes each subsystem's projected, semicanonical orbitals to its own Molden file, labelling the occupied ones as central or environmental whenever that distinction is available to the writer. `PRINT_LEVEL` raises the verbosity of the domain- and subsystem-construction bookkeeping printed along the way.

13.6 CIM Module Options

- **CIM_TYPE**
  Orbital-domain construction scheme.
  TYPE: String
  DEFAULT: SECIM
  OPTIONS: SECIM, ICIM

- **MODE**
  Controls how the domains are built and the correlation calculation is run.
  TYPE: String
  DEFAULT: SERIAL
  OPTIONS: SERIAL, SOW, SUBSYSTEM

- **COLLAPSE**
  Merges an LMO domain that is wholly embedded in a larger one into that larger domain, keeping every active LMO central in exactly one surviving domain.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **ORB_TOL**
  Mulliken-charge threshold used, in SECIM, both to assign central LMOs to a non-hydrogen atom and to decide which atoms belong to an LMO's AO domain.
  TYPE: Double
  DEFAULT: 0.15
  OPTIONS: n User-defined value.

- **ENV_TOL**
  Fock-matrix threshold used in SECIM to add environmental LMOs to a domain of central LMOs.
  TYPE: Double
  DEFAULT: 1.0×10^{-3}
  OPTIONS: n User-defined value.

- **BUFF_DISTANCE**
  Radius, in Å, within which non-hydrogen buffer atoms are added around a SECIM subsystem to form its extended subsystem.
  TYPE: Double

DEFAULT: 4.0
OPTIONS: n User-defined value.

- **VIR_CUTOFF**
  Retention threshold for a SECIM subsystem's virtual orbitals: a virtual projected from the extended subsystem down onto the subsystem AO domain is kept only if its squared overlap with the once-projected orbital is at least $1-$ `VIR_CUTOFF`.
  TYPE: Double
  DEFAULT: 0.2
  OPTIONS: n User-defined value.

- **ICIM_TOL**
  Distance threshold, in Å, between central atoms below which an LMO is added to another LMO's MO domain in ICIM.
  TYPE: Double
  DEFAULT: 5.5
  OPTIONS: n User-defined value.

- **APAO_CUTOFF**
  Least-squares threshold used to grow an atom's approximate PAO (APAO) domain in ICIM.
  TYPE: Double
  DEFAULT: 0.05
  OPTIONS: n User-defined value.

- **ORTH_CUTOFF**
  Overlap-eigenvalue threshold below which a direction is discarded in the canonical orthogonalization of a subsystem's projected atomic orbitals, shared by SECIM and ICIM.
  TYPE: Double
  DEFAULT: 1.0×10^{-5}
  OPTIONS: n User-defined value.

- **DRY_RUN**
  Builds every subsystem's orbitals and integrals and prints the domain-size summary, but skips the correlation calculation itself.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **WRITE_LOCAL_MOLDEN**
  Writes the whole-molecule Boys-localized orbitals to a Molden file before any subsystem projection takes place.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **WRITE_SUBSYS_MOLDEN**
  Writes each subsystem's projected, semicanonical orbitals to its own Molden file.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **PRINT_LEVEL**
  Verbosity of the domain- and subsystem-construction output.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

13.7 Input Examples

*Single-point `cim-mp2/cc-pvdz` computation of pentadecane with SECIM:

```
$set_molecule
  0 1
  C   0.0000030000   0.4577640000   0.0000950000
  ...
  H  -9.8804240000   0.2325170000  -0.0002240000
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method cim-mp2
  reference rhf
  freeze_core true
$end_globals

$set_scf
  local_maxiter 600
$end_scf

$set_cim
  env_tol 0.002
  orb_tol 0.15
  vir_cutoff 0.2
  orth_cutoff 1e-5
$end_cim
```

`FREEZE_CORE TRUE` is required by SECIM; the same molecule with `method cim-ccsd(t)` in place of `cim-mp2` runs the coupled-cluster route through exactly the same SECIM domains.

*Single-point `cim-mp2/cc-pvdz` computation of the same molecule with ICIM:

```
$set_molecule
  0 1
  C   0.0000030000   0.4577640000   0.0000950000
  ...
  H  -9.8804240000   0.2325170000  -0.0002240000
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method cim-mp2
  reference rhf
  freeze_core false
$end_globals

$set_scf
  local_maxiter 600
$end_scf

$set_cim
  cim_type icim
$end_cim
```

Here `CIM_TYPE ICIM` switches on the improved domain construction of Sec. 13.4 and its two options, `ICIM_TOL` and `APAO_CUTOFF`, are left at their defaults; ICIM does not require `FREEZE_CORE TRUE`.

14 DEC: Divide-Expand-Consolidate Methods

Code Authors: Y. Alagöz.

14.1 Overview

The DEC module provides divide-expand-consolidate (DEC) local-correlation methods, introduced by Jørgensen and coworkers [Ziółkowski, Jansik, Kjærgaard, and Jørgensen, J. Chem. Phys. **133**, 014107 (2010); Kristensen, Ziółkowski, Jansik, Kjærgaard, and Jørgensen, J. Chem. Theory Comput. **7**, 1677 (2011); Høyvik, Kristensen, Jansik, and Jørgensen, J. Chem. Phys. **136**, 014105 (2012)]. The available methods are `dec-mp2` and `dec-ccsd`. Like the CIM module (Sec. 13), DEC reassembles a correlation energy from calculations run on small, localized fragments of the molecule, but it builds those fragments around individual

atoms rather than merged central/environment domains, and it localizes the virtual space directly instead of projecting it from PAOs.

14.2 Atomic Sites and Orbital Extents

The active occupied *and* virtual orbitals are localized together, with `LOCAL_TYPE` choosing between the Boys and Pipek–Mezey criteria. Every localized orbital, occupied or virtual, is assigned to an *atomic site*: the atom carrying its largest population, Mulliken or Löwdin according to `CHARGE_TYPE`, exactly as an LMO is assigned a central atom in ICIM (Sec. 13.4). Every atom that ends up owning at least one occupied orbital becomes a *fragment*. Each orbital also gets an *orbital extent*, an AO domain grown one atom at a time – always adding next the atom of its next- largest population – until the orbital, projected onto the accumulated domain, is reproduced to within `ORBITAL_EXTENT_TOL`; a fragment's own atomic extent is the union of the orbital extents of the orbitals sited on it.

14.3 Fragment and Pair Energies

A fragment's correlation energy is not evaluated in its own orbitals alone: an *energy space* – its own occupied orbitals together with the virtual orbitals of every atom within `R_EFF_VIR` – is embedded in a larger *amplitude space* – the occupied and virtual orbitals of every atom within `R_BUFF_OCC` and `R_BUFF_VIR`, respectively – so that the amplitudes feel a wider correlating environment than the one the energy is finally read off from. The fragment is then run, on its own reduced molecule and AO basis, through the same semicanonicalization and DFOCC correlation step the CIM module uses (Sec. 13), with the energy expression restricted to orbital pairs inside the energy space alone. The total correlation energy adds a pairwise correction on top of the fragment sum,

$$E_{\mathrm{corr}} = \sum_{A} E_A + \sum_{A<B} \left(E_{AB} - E_A - E_B \right) ,$$

where E_{AB} is the same kind of calculation run on the union of sites A and B; the correction is what a simple sum of independent fragments misses. With `LIMIT_PAIRS TRUE` (the default), only pairs of sites within `MAX_PAIR_DIST` are evaluated, more distant pairs being assumed to contribute negligibly.

14.4 DEC Module Options

- **LOCAL_TYPE**
  Localization criterion applied to the active occupied and virtual orbitals.
  TYPE: String
  DEFAULT: BOYS
  OPTIONS: BOYS, PIPEK-MEZEY

- **CHARGE_TYPE**
  Population analysis used to assign an orbital to its atomic site.
  TYPE: String
  DEFAULT: MULLIKEN
  OPTIONS: MULLIKEN, LOWDIN

- **ORBITAL_EXTENT_TOL**
  Least-squares threshold used to grow an orbital's AO extent.
  TYPE: Double
  DEFAULT: 0.01
  OPTIONS: n User-defined value.

- **R_EFF_VIR**
  Radius, in Å, within which a site's virtual orbitals are added to a fragment's energy space.
  TYPE: Double
  DEFAULT: 1.2
  OPTIONS: n User-defined value.

- **R_BUFF_OCC**
  Radius, in Å, within which a site's occupied orbitals are added to a fragment's amplitude space.
  TYPE: Double
  DEFAULT: 0.0
  OPTIONS: n User-defined value.

- **R_BUFF_VIR**
  Radius, in Å, within which a site's virtual orbitals are added to a fragment's amplitude space.
  TYPE: Double
  DEFAULT: 2.5
  OPTIONS: n User-defined value.

- **LIMIT_PAIRS**
  Restricts the pair-energy sum to site pairs within `MAX_PAIR_DIST`.
  TYPE: Boolean
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **MAX_PAIR_DIST**
  Distance, in Å, beyond which a pair of sites is skipped when `LIMIT_PAIRS` is on.
  TYPE: Double
  DEFAULT: 7.0
  OPTIONS: n User-defined value.

- **PRINT_LEVEL**
  Verbosity of the site-, extent-, and fragment-construction output.

TYPE: Integer
DEFAULT: 0
OPTIONS: n User-defined value.

14.5 Input Example

*Single-point `dec-mp2/cc-pvdz` computation of pentadecane:

```
$set_molecule
  0 1
  C   0.0000030000   0.4577640000   0.0000950000
  ...
  H  -9.8804240000   0.2325170000  -0.0002240000
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method dec-mp2
  reference rhf
  freeze_core true
$end_globals

$set_dec
  r_eff_vir 1.2
  r_buff_vir 2.5
  max_pair_dist 7.0
$end_dec
```

`method dec-ccsd` in place of `dec-mp2` runs the coupled-cluster route through the same fragments and pairs.

15 ML: Machine-Learning Models

Code Authors: Y. Alagöz.

15.1 Overview

The Ml library integrates data-driven models into `MacroQC`, providing tools for training and applying machine-learning models of molecular properties alongside the package's conventional electronic-structure methods; on a GPU-enabled build its training loop runs on the same accelerated back end as the tensor library. It is reached through `JOBTYPE ML` rather

than through a METHOD, and is independent of the usual reference/correlation machinery: a run only needs the $set_ml block (Sec. 15.5) and, for training or validation, a directory of QM9-style extended-XYZ files [Ramakrishnan, Dral, Rupp, and von Lilienfeld, Sci. Data **1**, 140022 (2014)] – a geometry followed by a comment line of whitespace-separated target properties, one file per molecule. MODE selects whether the run trains a model, applies a previously trained one, or does both in sequence, and MODEL_FILE is where the fitted model is written to, or read back from. Training and validation geometries always come from TRAIN_DIR and VALID_DIR; a prediction, in PREDICT or TRAIN_PREDICT mode, is made on the single molecule given in the job's own $set_molecule block instead.

15.2 Molecular Descriptors

FCNN and KRR (Sec. 15.3) both need the molecule reduced to a single fixed-length vector, and a raw list of Cartesian coordinates will not do: translating or rotating the molecule, or simply renumbering its atoms, leaves the property unchanged, so a usable descriptor has to be invariant to all three, on top of coping with molecules of different sizes. DESCRIPTOR chooses among four constructions built on the Coulomb matrix of Rupp, Tkatchenko, Müller, and von Lilienfeld [Phys. Rev. Lett. **108**, 058301 (2012)],

$$M_{ij} = \begin{cases} \frac{1}{2} Z_i^{2.4}, & i = j \\ \dfrac{Z_i Z_j}{\|\mathbf{r}_i - \mathbf{r}_j\|}, & i \neq j \end{cases} ,$$

whose diagonal is a rough per-atom energy and whose off-diagonal is the Coulomb repulsion between every pair of nuclei; the exponent 2.4 on the diagonal is an empirical fit, not a physical one.

- CM flattens the upper triangle of M (diagonal included) into a vector, zero-padded up to the atom count of the largest molecule in TRAIN_DIR. It depends on how the atoms happen to be numbered in the input file, which is otherwise arbitrary, so a network trained on it has to absorb that arbitrariness along with the chemistry.
- SORTED_CM removes most of that sensitivity for free: the atoms are reordered by decreasing row norm $\sum_j M_{ij}^2$ before M is flattened, which fixes the same molecule to (very nearly) the same vector regardless of the atom order in the input file.
- CM_EIGVAL keeps only the eigenvalue spectrum of M, zero-padded to the largest atom count. This is exactly permutation invariant, at the cost of the off-diagonal detail the eigenvectors discarded.
- BOB (bag of bonds) [Hansen, Biegler, Ramakrishnan, Pronobis, von Lilienfeld, Müller, and Tkatchenko, J. Phys. Chem. Lett. **6**, 2326 (2015)] sorts the off-diagonal entries of M into one bag per pair of elements,

$$\mathrm{bag}_{AB} = \left\{ \frac{Z_i Z_j}{\|\mathbf{r}_i - \mathbf{r}_j\|} \;:\; Z_i \in A,\; Z_j \in B \right\} ,$$

plus one bag per element for the diagonal, sorts every bag from largest value to smallest, and zero-pads each one to the largest count of that pair (or element) seen anywhere in `TRAIN_DIR`; the bag sizes are worked out automatically by scanning the training directory before the network is built. Because the sort happens *within* each bag, `BOB` is exactly permutation invariant like `CM_EIGVAL`, but, being grouped by bond type rather than collapsed into eigenvalues, it stays chemically interpretable.

15.3 Models

`MODEL` selects among three families. `FCNN` is a fully-connected network over the whole-molecule descriptor of Sec. 15.2, with `HIDDEN_LAYERS` and `ACTIVATION` setting its hidden architecture. `KRR` performs kernel ridge regression, with a Gaussian or Laplacian kernel (`KRR_KERNEL`, `KRR_SIGMA`) and ridge strength `KRR_LAMBDA`, over the same descriptor family; `MAX_TRAIN_SAMPLES` caps how many training molecules enter its kernel matrix, which scales quadratically with that number.

`BPNN` instead follows the atom-centered scheme of Behler and Parrinello [Phys. Rev. Lett. **98**, 146401 (2007)]: rather than one descriptor for the whole molecule, every atom i gets its own descriptor vector built from its local environment, which is fed to a sub-network shared by every atom of the same element (a separate set of weights per element, all using the same `HIDDEN_LAYERS`), and the molecular property is the size-consistent sum $E_{\mathrm{tot}} = \sum_i E_{Z_i}$ of the per-atom outputs; summing already-local, element-resolved contributions is what makes `BPNN` both permutation invariant and applicable to a molecule of any size, unlike the fixed-length descriptors above. `ATOMIC_REF` subtracts a fitted per-element reference energy before training, so the sub-networks learn the deviation from that reference rather than the raw atomic total. `DESCRIPTOR` is ignored here (with a warning if it was set) – the atomic descriptor is fixed to the atom-centered symmetry functions (ACSF) of Behler [J. Chem. Phys. **134**, 074106 (2011)], in the ANI-1 form of Smith, Isayev, and Roitberg [Chem. Sci. **8**, 3192 (2017)]. Every neighbour j of atom i is smoothly switched off past a cutoff radius,

$$f_c(R) = \begin{cases} \frac{1}{2}\left[\cos(\pi R/R_c) + 1\right], & R \le R_c \\ 0, & R > R_c \end{cases},$$

and two families of symmetry functions accumulate over the neighbours that survive it: a radial function,

$$G^R_m = \sum_{j \neq i} \exp\left[-\eta\left(R_{ij} - R_s^{(m)}\right)^2\right] f_c(R_{ij}),$$

evaluated at 32 shells $R_s^{(m)}$ out to `BP_CUTOFF` and kept separate per element of j, and an angular function over neighbour pairs (j,k),

$$G^A_m = 2^{1-\zeta} \sum_{j,k \neq i} \left[1 + \cos(\theta_{ijk} - \theta_s)\right]^\zeta \exp\left[-\eta\left(\tfrac{R_{ij}+R_{ik}}{2} - R_s\right)^2\right] f_c(R_{ij}) f_c(R_{ik}),$$

swept over an 8×8 grid of angular shifts $\theta_s \in [0, \pi]$ and radial shifts R_s out to `BP_CUTOFF_ANGULAR`, and kept separate per pair of elements of j and k. Concatenating both gives every

atom's descriptor: for the five elements of QM9 (H, C, N, O, F) that is $5 \times 32 = 160$ radial and $15 \times 64 = 960$ angular features, 1120 in all, with the element set itself – and so the exact descriptor length – determined automatically from what `TRAIN_DIR` contains.

15.4 Training

Training minimizes the fit with `OPTIMIZER` (Adam or SGD) at `LEARNING_RATE`, which is reduced by `LR_FACTOR` whenever the validation loss stops improving for `LR_PATIENCE` epochs, down to `MIN_LR`; the run itself stops early once `PATIENCE` epochs pass without improvement, or at `MAX_EPOCH` regardless. A held-out validation set is read from `VALID_DIR` when given, or else split off automatically from the training directory according to `VALID_FRACTION`. `GRAD_CLIP` and `GRAD_CLIP_NORM` bound the gradient norm, and `WEIGHT_DECAY` adds L2 regularization; `RANDOM_SEED` fixes the split and the initial weights for a reproducible fit.

15.5 ML Module Options

- **MODEL**
  Model family to train or apply.
  TYPE: String
  DEFAULT: FCNN
  OPTIONS: FCNN, BPNN, KRR

- **MODE**
  Whether the run trains a model, applies one, or both.
  TYPE: String
  DEFAULT: TRAIN_PREDICT
  OPTIONS: TRAIN_PREDICT, TRAIN, PREDICT

- **TRAIN_DIR**, **VALID_DIR**
  Directories of QM9-style extended-XYZ files used for training and, optionally, validation; a value containing a path should be given in double quotes.
  TYPE: Line
  DEFAULT: none
  OPTIONS: a directory path.

- **MODEL_FILE**
  File a trained model is written to, or read back from.
  TYPE: Line
  DEFAULT: model.libml
  OPTIONS: a file path.

- **DESCRIPTOR**
  Whole-molecule descriptor used by `FCNN` and `KRR` (Sec. 15.2).
  TYPE: String

DEFAULT: SORTED_CM
OPTIONS: SORTED_CM, CM, BOB, CM_EIGVAL

- **TARGET_INDEX**
  Index, or indices, of the property-line field(s) to regress against (Sec. 15.6).
  TYPE: Integer array
  DEFAULT: [0]
  OPTIONS: n User-defined value(s).

- **HIDDEN_LAYERS**, **ACTIVATION**
  Hidden-layer sizes and activation function of the `FCNN` and `BPNN` sub-networks.
  TYPE: Integer array / String
  DEFAULT: $[128, 64]$ / TANH
  OPTIONS for `ACTIVATION`: TANH, RELU, SIGMOID, LINEAR, CELU, GAUSSIAN.

- **OPTIMIZER**, **LEARNING_RATE**
  Optimizer and initial learning rate.
  TYPE: String / Double
  DEFAULT: ADAM / 1.0×10^{-3}
  OPTIONS for `OPTIMIZER`: ADAM, SGD.

- **MAX_EPOCH**, **BATCH_SIZE**, **PATIENCE**
  Maximum number of training epochs, batch size, and early-stopping patience.
  TYPE: Integer
  DEFAULT: 1000 / 1 / 50
  OPTIONS: n User-defined value.

- **LR_PATIENCE**, **LR_FACTOR**, **MIN_LR**
  Plateau-based learning-rate schedule (Sec. 15.4).
  TYPE: Integer / Double / Double
  DEFAULT: 20 / 0.5 / 1.0×10^{-6}
  OPTIONS: n User-defined value.

- **GRAD_CLIP**, **GRAD_CLIP_NORM**, **WEIGHT_DECAY**
  Gradient-norm clipping and its threshold, and the L2 weight-decay coefficient.
  TYPE: Boolean / Double / Double
  DEFAULT: TRUE / 1.0 / 0.0
  OPTIONS: n User-defined value.

- **VALID_FRACTION**, **RANDOM_SEED**
  Fraction of `TRAIN_DIR` held out for validation when `VALID_DIR` is not given, and the random seed for that split and for weight initialization.
  TYPE: Double / Integer
  DEFAULT: 0.2 / -1 (random)
  OPTIONS: n User-defined value.

- **BP_CUTOFF**, **BP_CUTOFF_ANGULAR**, **ATOMIC_REF**
  Radial and angular cutoffs, in Å, of the `BPNN` symmetry functions (Sec. 15.3), and

whether a per-element reference is subtracted before fitting.
TYPE: Double / Double / Boolean
DEFAULT: 4.6 / 3.1 / TRUE
OPTIONS: n User-defined value; TRUE, FALSE.

- **KRR_KERNEL**, **KRR_SIGMA**, **KRR_LAMBDA**
  Kernel type, width, and ridge regularization strength for `KRR`.
  TYPE: String / Double / Double
  DEFAULT: GAUSSIAN / 1.0 / 1.0×10^{-4}
  OPTIONS for `KRR_KERNEL`: GAUSSIAN, LAPLACIAN.

- **MAX_TRAIN_SAMPLES**
  Caps the number of training molecules used by `KRR`, whose kernel matrix scales quadratically with that number.
  TYPE: Integer
  DEFAULT: -1 (unlimited)
  OPTIONS: n User-defined value.

- **PRINT_LEVEL**
  Verbosity of the training and prediction output.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

15.6 Input Example

QM9 stores its 15 target properties – among them the atomization-relevant total energies U_0, U, H, and G, the HOMO–LUMO gap, the dipole moment, and the heat capacity – as one whitespace-separated comment line per molecule, and `TARGET_INDEX` is simply the (zero-based) position of the property wanted in that line, counted from 0 and marked in red below the corresponding word. Rather than crop it down to that one line, `dsgdb9nsd_000001.xyz` – methane, the smallest molecule in the set – is reproduced here in full: the atom count, the comment line with `TARGET_INDEX [12]` marked, the five atom lines (element, x, y, z, and a Mulliken charge QM9 appends as a fifth column), the harmonic frequencies, and the GDB9/relaxed-geometry SMILES and InChI strings that close the file.

```
5
gdb 1 157.7118 157.70997 157.70699 0. 13.21 -0.3877 0.1171 0.5048 35.3641 0.044749 -40.47893 -40.476062 -40.475117 -40.498597 6.469
 0  1     2         3         4     5   6      7      8      9      10       11        12         13         14         15       16
C  -0.0126981359   1.0858041578   0.0080009958  -0.535689
H   0.002150416   -0.0060313176   0.0019761204   0.133921
H   1.0117308433   1.4637511618   0.0002765748   0.133922
H  -0.540815069    1.4475266138  -0.8766437152   0.133923
H  -0.5238136345   1.4379326443   0.9063972942   0.133923
1341.307  1341.3284  1341.365  1562.6731  1562.7453  3038.3205  3151.6034  3151.6788  3151.7078
C          C
InChI=1S/CH4/h1H4          InChI=1S/CH4/h1H4
```

The words of the comment line are the `gdb` tag and the dataset index (0–1); the rotational constants A, B, C (2–4); the dipole moment μ and the polarizability α (5–6); $\varepsilon_{\mathrm{HOMO}}$, $\varepsilon_{\mathrm{LUMO}}$,

and their gap (7–9); $\langle R^2 \rangle$ and the zero-point energy (10–11); and, boxed above, the internal energy U_0 at 0 K, the internal energy U, the enthalpy H, and the free energy G, all at 298 K (12–15), followed by the heat capacity C_v (16). `TARGET_INDEX [12]` is therefore U_0, the 0 K electronic-structure energy and the property `BPNN` models most naturally.

Training a `bpnn` model against U_0 over a local copy of QM9, then applying it to methane itself:

```
$set_molecule
  0 1
  C    0.000000    0.000000    0.000000
  H    0.629118    0.629118    0.629118
  H   -0.629118   -0.629118    0.629118
  H   -0.629118    0.629118   -0.629118
  H    0.629118   -0.629118   -0.629118
$end_molecule

$set_globals
  jobtype ml
$end_globals

$set_ml
  model         bpnn
  mode          train_predict
  train_dir     "/path/to/QM9"
  target_index  [12]
  model_file    "model.libml"
$end_ml
```

With `MODE TRAIN_PREDICT`, the fit runs first, entirely over the molecules in `TRAIN_DIR`, and the prediction that follows uses the methane geometry above – not a molecule drawn from `TRAIN_DIR` – exactly as `MODE PREDICT` alone would once `MODEL_FILE` already holds a trained model.

16 QDPT: Quasidegenerate Perturbation Theory

Code Author: U. Bozkaya. The `qdpt` module includes CAS-CI, QDPT2, and FCI methods. Publications resulting from the use of `qdpt` module should cite Bozkaya [19].

16.1 Input Examples

*Single point `qdpt2/cc-pvdz` computation with `cms(2,1,1)`, which corresponds to `cas(2,2)` active space:

```
$set_molecule H2O
```

```
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    method qdpt2
    reference uhf
    jobtype energy
    freeze_core false
$end_globals

$set_qdpt
    qdpt_maxiter 50
    nactmo 2
    nalpha 1
    nbeta 1
$end_qdpt
```

*Single point `cas-ci/cc-pvdz` computation with `cms(2,1,1)`, which corresponds to `cas(2,2)` active space:

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
    method cas-ci
    reference uhf
    jobtype energy
    freeze_core false
$end_globals

$set_qdpt
    qdpt_maxiter 50
    nactmo 2
    nalpha 1
    nbeta 1
$end_qdpt
```

*Single point `fci/cc-pvdz` computation:

```
$set_molecule H2O
    0 1
    B   0.00000   0.00000   0.00000
    H   0.00000   0.00000   1.00000
$end_molecule

$set_globals
    basis cc-pvdz
    method fci
    reference uhf
    jobtype energy
    freeze_core false
    memory 6000
$end_globals

$set_qdpt
    qdpt_maxiter 50
$end_qdpt
```

16.2 QDPT Module Options

- **DAVIDSON_TOLERANCE**
  Convergence criterion for Davidson (residuals).
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined tolerance value.

- **E_CONVERGENCE**
  Convergence criterion for energy.
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined value.

- **ON_MAX**
  Maximum value for occupation number cutoff.
  TYPE: Double
  DEFAULT: 1.98
  OPTIONS: n User-defined value.

- **ON_MIN**
  The minimum value for occupation number cutoff.
  TYPE: Double

DEFAULT: 0.02
OPTIONS: n User-defined value.

- **REG_PARAM**
  Regularization parameter.
  TYPE: Double
  DEFAULT: 0.4
  OPTIONS: n User-defined value.

- **SWAP_MO_INITIAL**
  This option must be used along with the SWAP_MO_FINAL option. SWAP_MO_INITIAL defines a vector for initial MO ordering. The vector should include the range of MOs that will be re-ordered.
  TYPE: Integer Array
  DEFAULT: [0]
  OPTIONS: User-defined.

- **SWAP_MO_FINAL**
  This option must be used along with the SWAP_MO_INITIAL option. SWAP_MO_FINAL defines a vector for final MO ordering.
  TYPE: Integer Array
  DEFAULT: [0]
  OPTIONS: User-defined.

- **CUTOFF**
  The cutoff value for numerical procedures.
  TYPE: Integer
  DEFAULT: 8
  OPTIONS: n User-defined value.

- **EXTERNAL_NFRZV**
  Externally defined frozen virtuals.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **FOLLOW_CI_ROOT**
  The electronic root that will be followed in a FCI computation.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **FOLLOW_ROOT**
  The electronic root that will be followed in a QDPT computation.
  TYPE: Integer

DEFAULT: 1
OPTIONS: n User-defined value.

- **NACTMO**
  Number of active MOs in the model space.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: n User-defined value.

- **NALPHA**
  Number of active alpha electrons in the model space.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **NBETA**
  Number of active beta electrons in the model space.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **NSWAPMO**
  Number of MOs to be swapped in the initial guess.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **PRINT_LEVEL**
  The amount of information to print to the output file.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **QDPT_MAXITER**
  Maximum number of iterations.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **CIVECS_READ**
  Do read CI coefficients from external files?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **CIVECS_WRITE**
  Do write CI coefficients to external files?

TYPE: Boolean
DEFAULT: FALSE
OPTIONS: TRUE, FALSE

- **CI_VECS**
  Do print CI coefficients?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **COMPUTE_S2**
  Do compute spin expectation value?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **CS_SINGLET**
  Do use closed-shell singlet wavefunction?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **FOCK_TYPE**
  Type of the Fock.
  TYPE: String
  DEFAULT: CORE
  OPTIONS: CORE, UHF, MCSCF, CASCI

- **FOLLOW_STATE**
  Follow what state, default is the ground state.
  TYPE: String
  DEFAULT: GROUND
  OPTIONS: GROUND, SINGLET, TRIPLET

- **MODEL_SPACE**
  How to select model space. It can be automatically determined via UNO or it can be defined by users.
  TYPE: String
  DEFAULT: USER
  OPTIONS: USER, UNO

- **MO_READ**
  Do read MO coefficients from external files?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **MO_WRITE**
  Do write MO coefficients to external files?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **NAT_ORBS**
  Do use MP2 NOs?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **OS_SINGLET**
  Do use open-shell singlet wavefunction?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **PRINT_CONFIGS**
  Do print configurations?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **REGULARIZATION**
  Do use regularized denominators?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **SC_MO**
  Do semi-canonicalize ROHF MOs?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **SYMMETRIC_HEFF**
  Do symmetrize effective Hamiltonian?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **UNO**
  Do form UHF natural orbitals?
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **REF_WFN_TYPE**
  Type of the ref wavefunction.
  TYPE: String
  DEFAULT: SCF
  OPTIONS: SCF

17 DFT: Kohn–Sham Density Functional Theory

Code Author: Y. Alagöz.

17.1 Overview

`MacroQC` performs Kohn–Sham density functional theory calculations for restricted and unrestricted references (RKS and UKS). A calculation is requested by setting the reference to `RKS` or `UKS` and naming a functional:

```
$set_globals
  reference rks
$end_globals

$set_scf
  dft_functional b3lyp
$end_scf
```

The exchange–correlation energy and potential are evaluated by numerical quadrature on atom-centered grids, partitioned between atoms by a fuzzy Voronoi scheme. Everything else — the guess, the DIIS, the integral back ends — is shared with the Hartree–Fock code of Section 8, so the same `SCF_TYPE` choices apply. An RKS calculation requires a closed-shell multiplicity; `MacroQC` stops rather than silently averaging.

17.2 Functionals

The functionals below are available. Local and gradient-corrected functionals cost only the quadrature; global hybrids additionally need exact exchange, and range-separated hybrids need it with an attenuated kernel, which is why they are the most expensive.

Class	Keyword	Description
exchange only	`slater`	Slater (Dirac) local exchange
	`b88`	Becke 1988 gradient-corrected exchange
	`pbex`	PBE exchange
correlation only	`vwn5`, `vwn3`	Vosko–Wilk–Nusair local correlation
	`lypc`	Lee–Yang–Parr correlation
	`pbec`	PBE correlation
LDA	`svwn`	Slater exchange with VWN correlation
GGA	`blyp`	B88 exchange with LYP correlation
	`slyp`	Slater exchange with LYP correlation
	`spbec`	Slater exchange with PBE correlation
	`pbe`	PBE exchange and correlation
meta-GGA	`m06l`	M06-L
global hybrid	`b3lyp`	B3LYP, 20% exact exchange
	`pbe0`	PBE0, 25% exact exchange
	`b97`	B97
	`m06`	M06, 27% exact exchange
	`m06-2x`	M06-2X, 54% exact exchange
range-separated	`wb97`	ωB97
	`wb97x`	ωB97X

17.3 The Integration Grid

The exchange–correlation integrals are evaluated on a product grid: a radial quadrature per atom, a Lebedev angular grid on each radial shell, and a nuclear partition function that divides space between the atoms. The defaults are accurate enough for energies and gradients of ordinary organic molecules; a meta-GGA or a functional with a steep enhancement factor may need a finer grid, which is what `NUM_RADIAL` and `LEBEDEV_L` are for.

17.3.1 NUMQUAD Module Options

- **NUM_RADIAL**
  TYPE: INTEGER
  DEFAULT: 75
  DESCRIPTION: the number of radial shells per atom. The cost of the quadrature is linear in it.

- **LEBEDEV_L**
  TYPE: INTEGER
  DEFAULT: 29
  DESCRIPTION: the order of the Lebedev grid on each shell; order 29 carries 302 points.

- **QUADRATURE**
  TYPE: STRING
  DEFAULT: GC
  OPTIONS: GC, DE
  DESCRIPTION: the radial quadrature rule — Gauss–Chebyshev of the second kind, or the double-exponential rule.

- **MAPPING**
  TYPE: STRING
  DEFAULT: TA
  OPTIONS: TA, DE2
  DESCRIPTION: how the quadrature interval is mapped onto the radial coordinate. TA is the Treutler–Ahlrichs M4 mapping, which places more points where the density varies fastest.

- **TA_ALPHA**
  TYPE: DOUBLE
  DEFAULT: 0.6
  DESCRIPTION: the exponent of the M4 mapping.

- **NUCLEAR_SCHEME**
  TYPE: STRING
  DEFAULT: TREUTLER
  OPTIONS: TREUTLER, BECKE, SSF
  DESCRIPTION: the fuzzy partition of space between atoms. TREUTLER adjusts the cell boundaries for atomic size, BECKE is the original scheme, and SSF is the faster stratified variant.

- **PRUNING**
  TYPE: STRING
  DEFAULT: NONE
  OPTIONS: NONE, TREUTLER
  DESCRIPTION: whether to lower the angular order on the innermost and outermost radial shells, where a full grid is wasted.

- **WEIGHTS_TOL**
  TYPE: DOUBLE
  DEFAULT: 10^{-15}
  DESCRIPTION: grid points whose weight falls below this are discarded.

17.4 Chain-of-Spheres Exchange

For a hybrid functional the exact-exchange matrix is usually the bottleneck. The chain-of-spheres (COSX) approximation evaluates it semi-numerically, on the same kind of grid the functional uses, which is much faster than assembling it from four-index integrals. It is

switched on with `COSX TRUE` in the `SCF` block, and it uses a coarse grid while the SCF is still far from convergence and a finer one for the last iterations.

17.4.1 COSX Options (SCF block)

- **COSX**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: evaluate the exact-exchange matrix semi-numerically.
- **COSX_RADIAL_INITIAL**, **COSX_LEBEDEV_INITIAL**
  TYPE: INTEGER
  DEFAULT: 25 and 11
  DESCRIPTION: the coarse grid used while the density is still changing.
- **COSX_RADIAL_FINAL**, **COSX_LEBEDEV_FINAL**
  TYPE: INTEGER
  DEFAULT: 35 and 17
  DESCRIPTION: the finer grid used for the converged density.
- **COSX_MAXITER_FINAL**
  TYPE: INTEGER
  DEFAULT: 1
  DESCRIPTION: how many iterations are run on the final grid.
- **COSX_OVERLAP_FITTING**
  TYPE: BOOL
  DEFAULT: TRUE
  DESCRIPTION: correct the numerical exchange matrix with the analytic overlap, which removes most of the grid error at negligible cost.
- **COSX_PRUNING_INITIAL**, **COSX_PRUNING_FINAL**
  TYPE: STRING
  DEFAULT: NONE
  DESCRIPTION: pruning schemes for the two COSX grids, independent of the one the functional uses.
- **COSX_INTS_TOLERANCE**
  TYPE: DOUBLE
  DEFAULT: 10^{-11}
  DESCRIPTION: the screening threshold of the collocation matrices.

17.5 Empirical Dispersion

Semilocal functionals do not describe London dispersion, and an empirical correction is the cheapest remedy. It depends on the nuclear positions alone, so it costs nothing and

contributes to the gradient analytically.

- **EMPIRICAL_DISPERSION**
  TYPE: STRING
  DEFAULT: NONE
  OPTIONS: NONE, GD2, GD3, GD3BJ
  DESCRIPTION: Grimme's D2 correction, D3 with zero damping, or D3 with Becke–Johnson damping. GD3BJ is the usual recommendation.

18 CIS and TDA/TD-DFT: Excited States

Code Author: U. Bozkaya.

18.1 Overview

Configuration interaction singles (CIS) is the simplest wave-function description of an excited state: the excited-state wave function is a linear combination of all single excitations out of the reference determinant, and the excitation energies are the eigenvalues of

$$A_{ia,jb} \;=\; \delta_{ij}\delta_{ab}\,(\epsilon_a - \epsilon_i) \;+\; \langle aj||ib\rangle. \tag{39}$$

Applied to a Kohn–Sham reference with the exchange–correlation kernel in place of the exchange integrals, the same equation is the Tamm–Dancoff approximation to TD-DFT. Dropping the approximation adds the coupling block B and gives full linear response.

An excited-state calculation is requested on top of an SCF reference:

```
$set_globals
  method cis
$end_globals

$set_cis
  ee_singlets 5
  ee_triplets 3
$end_cis
```

For an RHF reference the singlet and triplet manifolds decouple and are found in two separate passes, which is why they are requested by separate counts rather than by one number of roots. For a UHF reference the two are mixed and only `EE_SINGLETS` is read.

18.2 Algorithms

The cost of CIS is dominated by one operation, the product of the CIS matrix with a trial vector, and `MacroQC` offers several ways to form it. The choice does not change the answer, only the time and the memory.

CIS_ALGORITHM	AO_JK_TYPE	Description
ao_basis	df_direct	three-index integrals rebuilt for every sigma; no large storage. The default.
ao_basis	df_incore	three-index integrals read from disk.
ao_basis	conv_eri_direct	four-index integrals computed on the fly; no auxiliary basis is touched at all.
ao_basis	conv_eri_incore	four-index integrals built once, packed, and contracted from memory; the fastest where it fits.
mo_basis	—	the CIS matrix is built in the molecular-orbital basis.

18.3 CIS Module Options

- **EE_SINGLETS**
  TYPE: INTEGER
  DEFAULT: 4
  DESCRIPTION: the number of singlet excited states to find. For a UHF reference this is simply the number of states.

- **EE_TRIPLETS**
  TYPE: INTEGER
  DEFAULT: 0
  DESCRIPTION: the number of triplet excited states, for a closed-shell reference only. Singlets and triplets are found in separate passes.

- **CIS_ALGORITHM**
  TYPE: STRING
  DEFAULT: AO_BASIS
  OPTIONS: AO_BASIS, MO_BASIS
  DESCRIPTION: whether the matrix-vector product is formed in the atomic or the molecular orbital basis.

- **AO_JK_TYPE**
  TYPE: STRING
  DEFAULT: DF_DIRECT
  OPTIONS: DF_DIRECT, DF_INCORE, CONV_ERI_DIRECT, CONV_ERI_INCORE
  DESCRIPTION: which integrals the AO-basis product is built from and whether they are stored. The names say both: `df_` means density fitting and `conv_eri_` the conventional four-index integrals, while `_direct` means recomputed and `_incore` means kept.

- **DIAG_METHOD**
  TYPE: STRING
  DEFAULT: DAVIDSON
  OPTIONS: DAVIDSON, FULL_DIAG
  DESCRIPTION: iterative subspace diagonalization, or an explicit diagonalization of the whole matrix. The latter is only practical for small systems and is mainly useful as a check.

- **VECS_PER_ROOT**
  TYPE: INTEGER
  DEFAULT: 1
  DESCRIPTION: how many trial vectors are started per root. Raising it helps when roots are nearly degenerate.

- **CIS_E_CONV**
  TYPE: DOUBLE

DEFAULT: 10^{-6}
DESCRIPTION: convergence threshold on the excitation energy.

- **CIS_R_CONV**
  TYPE: DOUBLE
  DEFAULT: 10^{-6}
  DESCRIPTION: convergence threshold on the norm of the residual vector.

- **DAVIDSON_TOLERANCE**
  TYPE: DOUBLE
  DEFAULT: 10^{-6}
  DESCRIPTION: the threshold below which a new expansion vector is considered linearly dependent and discarded.

- **CIS_GUESS_COEFF**
  TYPE: STRING
  DEFAULT: MIX
  OPTIONS: MIX, SINGLET, TRIPLET
  DESCRIPTION: the spin character of the starting vectors.

- **RHF_CIS_STATE**
  TYPE: STRING
  DEFAULT: SINGLET
  OPTIONS: SINGLET, TRIPLET
  DESCRIPTION: which manifold a single pass targets; the driver sets it from `EE_SINGLETS` and `EE_TRIPLETS`.

- **FOLLOW_ROOT**
  TYPE: INTEGER
  DEFAULT: 1
  DESCRIPTION: which root a property or a subsequent step refers to.

- **EXCITATION_RANGE**
  TYPE: INTEGER
  DEFAULT: 2
  DESCRIPTION: how far above the requested number of roots the subspace is allowed to reach while converging.

- **CUTOFF**, **INTEGRAL_CUTOFF**
  TYPE: INTEGER
  DEFAULT: 8 and 9
  DESCRIPTION: negative powers of ten used as screening thresholds.

18.4 TDA and TD-DFT

With a Kohn–Sham reference the same machinery gives the Tamm–Dancoff approximation, and dropping that approximation gives full linear response. Both are requested from the `SCF`

block:

- **TD**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: run a linear-response calculation on top of the converged SCF reference.

- **TDA**
  TYPE: BOOL
  DEFAULT: TRUE
  DESCRIPTION: use the Tamm–Dancoff approximation, which keeps only the A block. Setting it to false includes the coupling block B and gives full linear response.

The Tamm–Dancoff approximation is not merely cheaper: it is markedly more robust near a conical intersection or wherever the reference is close to a triplet instability, where full linear response can return an imaginary excitation energy. With a Hartree–Fock reference TDA is exactly CIS.

19 EOMEE : Equation-of-Motion Coupled-Cluster

Code Authors: A. Ünal and U. Bozkaya. `MacroQC` includes EOM-CCD, EOM-OCCD, and EOM-CCSD implementations with the density-fitting approach [71, 72, 73, 74].

19.1 Input Examples

*Single point `eom-ccsd/cc-pvdz` computation for 5 roots:

```
$set_molecule H2O
  0 1
  O    0.00000    0.00000   -0.06577
  H    0.00000   -0.75906    0.52195
  H    0.00000    0.75906    0.52195
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method eom-ccsd
  reference rhf
  jobtype energy
  freeze_core true
$end_globals

$set_eomee
  num_roots 5
$end_eomee
```

19.2 EOMEE Module Options

- **NUM_ROOTS**
  Number of excited states for EOM computations.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **EXCITATION_RANGE**
  The gap between the occupied and virtual spaces from which initially excited state guesses are directed to the Davidson algorithm. The default value of 2 corresponds to HOMO-1, HOMO, LUMO, and LUMO+1.
  TYPE: Integer

DEFAULT: 2
OPTIONS: n User-defined value.

- **EOM_MAXITER**
  The maximum number of EOM iterations.
  TYPE: Integer
  DEFAULT: 80
  OPTIONS: n User-defined value.

- **MAX_VECS**
  Maximum dimension of the subspace.
  TYPE: Integer
  DEFAULT: 50
  OPTIONS: n User-defined value.

- **VECS_PER_ROOT**
  Subspace vectors used per root.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: n User-defined value.

- **SCHMIDT_ADD_RES_TOL**
  Tolerance value to add a guess vector of a root to the Davidson algorithm.
  TYPE: Double
  DEFAULT: 1.0e-3
  OPTIONS: n User-defined tolerance value.

- **EOM_E_CONVERGENCE**
  Convergence criterion for excitation energy.
  TYPE: Double
  DEFAULT: 1.0e-6
  OPTIONS: n User-defined tolerance value.

- **EOM_RES_PRECONDITION_TOL**
  Tolerance value of EOM residual preconditioning.
  TYPE: Double
  DEFAULT: 1.0e-4
  OPTIONS: n User-defined tolerance value.

- **EOMCC_GUESS**
  Specifies the initial guess of EOM-CC computations.
  TYPE: String
  DEFAULT: CIS
  OPTIONS: CIS, UNIT

  – CIS: The initial guess is generated from the configuration interaction singles method.

- UNIT: The initial guess is generated from unit vectors.

- **UNIT_GUESS_TYPE**
  Type of initial guesses.
  TYPE: String
  DEFAULT: GEN
  OPTIONS: LIMITED, GEN

  - LIMITED: This option requests a limited guess to be generated around HOMO-LUMO orbitals. The dimension of guess is determined by the EXCITATION_RANGE option.
  - GEN: All singly excited Slater determinants are used in generating an initial guess.

- **EOM_UNIT_GUESS_COEFF**
  Initial guess for unrestricted EOM computations.
  TYPE: String
  DEFAULT: MIX
  OPTIONS: MIX, SINGLET, TRIPLET

  - MIX: Use a mixture of singlet and triplet states.
  - SINGLET: Use singlet coefficients, which means $c_I^A = 1/\sqrt{2}$ and $c_i^a = 1/\sqrt{2}$.
  - TRIPLET: Use triplet coefficients, which means $c_I^A = 1/\sqrt{2}$ and $c_i^a = -1/\sqrt{2}$.

- **SYMMETRIC_HEFF**
  Symmetrize the effective Hamiltonian.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **PPL_TYPE**
  Type of the EOM-CCSD PPL term.
  TYPE: String
  DEFAULT: AUTO
  OPTIONS: AUTO, CD

- **CD_PRESCREEN**
  Do apply prescreening for the CD algorithm? This option can be used when the ppl_type is cd.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

20 Fragment: Molecular Fragmentation Approaches

Code Authors: U. Bozkaya and B. Ermiş.

20.1 FRAGMENT Module Options

- **FRAG_METHOD**
  Type of fragment methods.
  TYPE: String
  DEFAULT: LSSMF
  OPTIONS: LSSMF, NBODY, MFCC

- **MODE**
  Type of the mode.
  TYPE: String
  DEFAULT: RUN
  OPTIONS: RUN, SOW, REAP, SPLIT

  - SOW : Forms fragment files for mode sow computations.
    * If frag_method is LSSMF : For `nb_level 1`, `bonded.inp`, `nonbonded.inp`, and `groups.inp` files are written. For `nb_level 2` and `nb_level 3`, `bonded.inp`, `dimers.inp`, and `monomers.inp` files are written. In case of a large number of fragments, users may further split each file into the desired number of input files. For this purpose, the `npiece_b`, `npiece_nb`, `npiece_g`, `npiece_d`, and `npiece_m` options can be used.
    * If frag_method is NBODY : Forms nbody files for mode sow computations. `nbody.inp` files are written.
    * If frag_method is MFCC : Forms MFCC fragment files for mode sow computations. `capped_frags.inp`, `concaps.inp`, `capped_frags_ligand.inp`, `concap_ligand.inp`, and `ligand.inp` files are written.
  - REAP : When computations are completed for fragments generated with the `SOW` option, `REAP` reads output files to compute fragment energies. Note that when `MacroQC` is executed with the `REAP` option, all required output files should be available in the working directory.
  - RUN : The fragments are computed by this run rather than written out for another one. They are generated on the fly and a single output file is formed. Whether they are computed one after another or in parallel is set by `FRAG_NPROC` and, across nodes, by MPI — see section 5.4; `MODE` says who does the work and not how many of them there are.

    This value used to be spelled `SERIAL`, which named an execution style rather than a destination and became misleading once the fragments could be run in parallel. `SERIAL` is still accepted, so existing input files need no change.

- SPLIT : Splits fragment input files created with the SOW option into the given number of files.
  - If frag_method is LSSMF : This option should be used with the LSSMF npiece_b, npiece_nb, npiece_g, npiece_d, and npiece_m options.
  - If frag_method is NBODY : This option should be used with the N-Body npiece options.
  - If frag_method is MFCC : This option should be used with the MFCC npiece option.

- **SOFTWARE**
  For which quantum chemistry software input files will be prepared? This option must be used along with the MODE SOW and MODE REAP options. The available software choices depend on FRAG_METHOD. For LSSMF, the supported options are MACROQC, PSI4, QCHEM, and ORCA. For NBODY, the supported options are MACROQC and ORCA. For MFCC, the current implementation supports MACROQC.
  TYPE: String
  DEFAULT: MACROQC
  OPTIONS: MACROQC, PSI4, QCHEM, ORCA

- **FRAG_JOBTYPE**
  What is computed for each fragment. ENERGY is the default. GRAD makes the fragment calculations produce gradients, which are assembled into the gradient of the whole molecule; this is what a fragmented JOBTYPE GRAD, OPT, FREQ, IRC, AIMD, FSM, GSM or NEB run needs, and it is available for LSSMF and MFCC in MODE SERIAL. A fragmented JOBTYPE FREQ is the finite-difference Hessian of that gradient; which of the two parallel layers – the displacements or the fragments – runs is chosen by the frequency module's PARALLEL_LAYER, and never both. EXCITATION requests excited states.
  TYPE: String
  DEFAULT: ENERGY
  OPTIONS: ENERGY, GRAD, EXCITATION

- **FRAG_NPROC**
  How many fragments to compute at the same time. The fragments of a phase do not depend on one another, so they can be run concurrently: each worker is a separate process with its own molecule, options and scratch files, and computes a share of the fragments. The number is a real count of simultaneous electronic structure calculations, so it should not exceed the cores available, and memory scales with it — the MEMORY option is per process. Where the fragment calculations themselves are threaded (through the BLAS, for instance), the two levels share one budget: FRAG_NPROC $\times$ threads per worker $\leq$ cores.

  It is refused, with the reason, wherever the fragments are not independent: outside MODE RUN (except an excited-state run with LOCAL_EXCITATION PRESCREENING, whose groups are computed in process whatever the mode), for SOFTWARE other than MACROQC, and with CFM. Excited states are divided like anything else under MODE RUN; under SOW or REAP there is nothing here to divide. The answers do not depend on it: the same

energies, gradients and excitation energies come out to the last digit whatever the worker count.
TYPE: Integer
DEFAULT: 1
OPTIONS: n User-defined value.

- **PRINT_LEVEL**
  The amount of information to print to the output file.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

20.2 Linear-Scaling Systematic Systematic Molecular Fragmentation (LSSMF)

The `MacroQC` software provides the LSSMF approach for all theoretical ground-state methods available in it. Further, our LSSMF code can also collaborate with several quantum chemistry software. The current version can be used with the Psi4 [53], Q-Chem [75], and ORCA [57] packages.

20.2.1 Input Examples

- **Serial Mode:**

This is an example of a single-point energy computation at the `LSSMF-rhf/cc-pvdz` level. In this example, `level 3` is used for generating bonded fragments. Nothing needs to be set for the functional groups: with the default `FUNCTIONAL_GROUPS TRUE` every carboxylic acid, ester, amide, acetal, disulfide, peroxide, nitro group and so on (see `FUNCTIONAL_GROUPS` below for the list) is recognised and kept in one group, and the output lists the groups found together with their charges and their deviation from the library's reference geometry:

```
Functional groups found: 4
  group                  atoms                  charge  max|
    dr|/A  max|dtheta|
  carboxylate            12,13,14                   -1
    0.026          3.9
  amide                  3,4,5                       0
    0.020          1.4
  amide                  8,9,10                      0
    0.043          1.0
  ammonium               1,2                         1
    0.009          0.0
```

(Gly-Ala-Gly as a zwitterion; the second atom of the ammonium is the carbon it hangs on, matched so that the C–N bond can be checked but not merged into the group.) To fragment

by the distance thresholds alone, as earlier versions did, add `functional_groups false` to `$set_lssmf`; to add groups of your own, point `fg_library_extra` at a file in the format of `share/macroqc/fragment/functional_groups.dat`.

```
# TEST_LABEL: 3-3-dimethylheptane_level3
$set_molecule
0 1
C       -2.701034000     -1.655296000      0.075154000
C       -2.373842000     -0.295418000     -0.551340000
C       -1.121132000      0.456476000     -0.019192000
C       -1.062210000      1.813812000     -0.747749000
C       -1.257793000      0.707881000      1.495575000
C        0.149643000     -0.378389000     -0.331976000
C        1.503136000      0.209803000      0.088913000
C        2.682769000     -0.702310000     -0.270142000
C        4.039004000     -0.133276000      0.152781000
H       -3.611969000     -2.075889000     -0.381179000
H       -1.895007000     -2.391312000     -0.074579000
H       -2.886260000     -1.579045000      1.158788000
H       -2.256461000     -0.424469000     -1.643066000
H       -3.246081000      0.370694000     -0.420386000
H       -0.218422000      2.431448000     -0.402749000
H       -0.957409000      1.677367000     -1.837538000
H       -1.985366000      2.391184000     -0.572273000
H       -0.459662000      1.368810000      1.868629000
H       -2.220489000      1.194614000      1.726545000
H       -1.208561000     -0.228561000      2.074156000
H        0.047882000     -1.369086000      0.145378000
H        0.172883000     -0.570617000     -1.421514000
H        1.659232000      1.194490000     -0.386834000
H        1.514669000      0.391461000      1.178685000
H        2.534735000     -1.691722000      0.201172000
H        2.682886000     -0.885237000     -1.360734000
H        4.865160000     -0.810408000     -0.118293000
H        4.233578000      0.839788000     -0.330129000
H        4.083420000      0.026781000      1.243777000
$end_molecule

$set_globals
    basis cc-pvdz
    aux_basis_scf cc-pvdz-jkfit
    method scf
    reference rhf
    jobtype energy
    mol_frag true
```

```
    memory 1000
$end_globals

$set_frag
  software macroqc
  mode run
  frag_method lssmf
$end_frag

$set_lssmf
    level 3
    nb_cutoff_max 10.0
    mode run
$end_lssmf
```

In this example, `mol_frag true` option request a single-point LSSMF-SCF energy computation. The `level 3` option in the `lssmf` block indicates that a level 3 fragmentation will be used for bonded fragments. The `nb_cutoff_max 10.0` option indicates a value of 10.0 Å will be used for Δ_{nb}. The `mode run` option indicates that this job computes its fragments itself rather than writing them out, and that a single output file will be formed. When we execute the above example, we will see the following options in the output file:

```
================================================================
Natom: 29
Ngroup: 9
Nbfrag: 26
Nnonbfrag: 10
Charge & Multiplicity: 0 1
Nelectron: 74
LSSMF level: 3
Nonbonded level: 1
Print level: 0
R12 tolerance: 10.00
Delta single bond: 0.40
Delta double bond: 0.07
Delta triple bond: 0.25
Group files will be splitted into 0 parts
Bonded files will be splitted into 0 parts
Nonbonded files will be splitted into 0 parts
Nuclear Repulsion Energy: 487.8480383549
================================================================
```

These are the basic information about the LSSMF approach, which means for the given molecule there are 9 groups, 26 bonded fragments, and 10 no-bonded fragments. This section also prints out the basic parameters that used in the LSSMF approach: level 3 is used for bonded fragments, level 1 is used for NB fragments, Δ_{nb} tolerance of 10.0 Å is used, etc.

When energies of all bonded fragments are completed, the following table is printed out. In this table, the label and the coefficient of each bonded group, as well as corresponding energies, are provided. At the end of the table, the total bonded energy is printed.

```
*****************************************************************
                          SCF ENERGY
*****************************************************************

Coeff |           Energy |    Coeff*Energy | Bonded Group Name |
------|------------------|-----------------|-------------------|
  1   | -157.3064946038 | -157.3064946038 |         G_1_2_3_4 |
  1   | -157.3055496928 | -157.3055496928 |         G_1_2_3_5 |
  1   | -157.3038834931 | -157.3038834931 |         G_1_2_3_6 |
  1   | -157.3061863028 | -157.3061863028 |         G_2_3_4_5 |
  1   | -157.3049150998 | -157.3049150998 |         G_2_3_4_6 |
  1   | -157.3052965489 | -157.3052965489 |         G_2_3_5_6 |
  1   | -157.3050898279 | -157.3050898279 |         G_2_3_6_7 |
  1   | -157.3069522452 | -157.3069522452 |         G_3_4_5_6 |
  1   | -157.3051381643 | -157.3051381643 |         G_3_4_6_7 |
  1   | -157.3057145289 | -157.3057145289 |         G_3_5_6_7 |
  1   | -157.3074706011 | -157.3074706011 |         G_3_6_7_8 |
  1   | -157.3078019201 | -157.3078019201 |         G_6_7_8_9 |
 -2   | -118.2701325572 |  236.5402651145 |           G_1_2_3 |
 -2   | -118.2684223683 |  236.5368447365 |           G_2_3_4 |
 -2   | -118.2687952758 |  236.5375905517 |           G_2_3_5 |
 -3   | -118.2675117406 |  354.8025352217 |           G_2_3_6 |
 -1   | -118.2706308152 |  118.2706308152 |           G_3_4_5 |
 -2   | -118.2691546284 |  236.5383092567 |           G_3_4_6 |
 -2   | -118.2695519038 |  236.5391038076 |           G_3_5_6 |
 -3   | -118.2703538608 |  354.8110615825 |           G_3_6_7 |
 -1   | -118.2698076315 |  118.2698076315 |           G_6_7_8 |
  3   |  -79.2318269995 | -237.6954809985 |             G_2_3 |
  1   |  -79.2336613594 |  -79.2336613594 |             G_3_4 |
  1   |  -79.2340541054 |  -79.2340541054 |             G_3_5 |
  3   |  -79.2325730649 | -237.6977191947 |             G_3_6 |
 -1   |  -40.1984657167 |   40.1984657167 |               G_3 |
-----------------------------------------------------------------
Total bonded energy for SCF = -352.4867942522
-----------------------------------------------------------------
```

Similarly, all nonbonded dimers, their monomers, and individual energies are printed out as below.

```
Energy of Dimer|Energy of Mono1|Energy of Mono2|Nonbonded I.E.|Nonbonded I.E.|Dimer Names|
---------------|---------------|---------------|--------------|--------------|-----------|
   (hartree)   |     (hartree) |     (hartree) |    (hartree) |   (kcal/mol) |           |
---------------|---------------|---------------|--------------|--------------|-----------|
  -80.39565616 |  -40.19782045 |  -40.19788569 |   0.00004998 |   0.03136212 |   G_1_G_7 |
  -80.39556601 |  -40.19782045 |  -40.19773805 |  -0.00000752 |  -0.00471729 |   G_1_G_8 |
```

```
  -80.39573075 |  -40.19782045 |  -40.19790816 |  -0.00000214 |  -0.00134471 |   G_1_G_9 |
  -80.39444135 |  -40.19672185 |  -40.19773805 |   0.00001855 |   0.01163874 |   G_2_G_8 |
  -80.39463707 |  -40.19672185 |  -40.19790816 |  -0.00000706 |  -0.00443044 |   G_2_G_9 |
  -80.39635460 |  -40.19846572 |  -40.19790816 |   0.00001928 |   0.01209736 |   G_3_G_9 |
  -80.39571052 |  -40.19802493 |  -40.19773805 |   0.00005245 |   0.03291412 |   G_4_G_8 |
  -80.39594110 |  -40.19802493 |  -40.19790816 |  -0.00000801 |  -0.00502865 |   G_4_G_9 |
  -80.39568849 |  -40.19800026 |  -40.19773805 |   0.00004982 |   0.03126424 |   G_5_G_8 |
  -80.39591563 |  -40.19800026 |  -40.19790816 |  -0.00000721 |  -0.00452506 |   G_5_G_9 |


Nonbonded interaction energies larger than 1.0 kcal/mol.
-------------------------------------------------------------------------
Sum of nonbonded interaction energies for SCF ==>         0.000158133758 (hartree)
0.1 (kcal/mol)
-------------------------------------------------------------------------
```

Finally, the total LSSMF energy is printed out separately as follows:

```
     Total energy ==>       -352.486636118422
```

- **Different levels of theory for the bonded and the nonbonded fragments:**

The nonbonded corrections are interaction energies, so they may be computed at another level than the bonded expansion (see `NB_METHOD`). Here the bonded fragments of n-heptane are computed at the MP2/cc-pVDZ level and the nonbonded pairs and their monomers at the SCF/cc-pVDZ level; `NB_BASIS`, `NB_AUX_BASIS_SCF` and `NB_AUX_BASIS_CORR` would change the basis in the same way, and the same input with `jobtype grad` gives the gradient assembled from the MP2 bonded and the SCF nonbonded gradients.

```
$set_molecule
0 1
C        3.2171710000       0.8842760000       0.2852810000
...
H       -3.9662530000      -0.9510910000      -0.2374760000
$end_molecule

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  jobtype energy
  method mp2
  nocom true
  mol_frag true
  memory 2500
$end_globals

$set_frag
  software macroqc
  mode run
```

```
  frag_method lssmf
$end_frag

$set_lssmf
  level 3
  nb_level 1
  nb_method scf
$end_lssmf
```

The output states the two levels once the options are read, and the energy table names each:

```
   Level of theory
     bonded fragments    : MP2/CC-PVDZ (RHF)
     nonbonded fragments : SCF/CC-PVDZ (RHF)
   ...
   ***********************************************************************
               MP2/CC-PVDZ (RHF) (bonded) + SCF/CC-PVDZ (RHF) (nonbonded) ENERGY
   ***********************************************************************

   Coeff |          Energy |    Coeff*Energy | Bonded Group Name |
   ------|-----------------|-----------------|-------------------|
       1 | -157.8967782577 | -157.8967782577 |         G_1_2_3_4 |
   ...
   ------------------------------------------------------------------
    Total bonded energy for MP2 = -275.4396407478
   ------------------------------------------------------------------
   ...
    Sum of nonbonded interaction energies for SCF ==>         0.000067768574 (hartree)
   ...
    Total energy ==>      -275.439572979187
```

The total is the MP2 bonded energy of the plain `LSSMF-MP2` run plus the nonbonded sum of the plain `LSSMF-SCF` run. The basis and the reference may change with the method; a B3LYP/cc-pVDZ backbone with MP2/aug-cc-pVTZ nonbonded corrections, gradient included, is

```
$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  method scf
  reference rks
  jobtype grad
  mol_frag true
$end_globals

$set_scf
  dft_functional b3lyp
$end_scf

$set_frag
  software macroqc
  mode run
  frag_method lssmf
```

```
  frag_jobtype grad
$end_frag

$set_lssmf
  level 4
  nb_level 2
  nb_method mp2
  nb_basis aug-cc-pvtz
  nb_aux_basis_scf aug-cc-pvtz-jkfit
  nb_aux_basis_corr aug-cc-pvtz-ri
  nb_reference rhf
  nb_dft_functional none
$end_lssmf
```

In `MODE SOW` the same options write `bonded.inp` with `METHOD MP2` and `groups.inp` and `nonbonded.inp` with `METHOD SCF`; the `REAP` input then carries `method_name [mp2]` and `nb_method_name [scf]`.

- **Mode Sow-1:**

This is an example of a single-point energy computation at the `LSSMF-ccsd(t)/cc-pvdz` level. In this example, `level 3` is used for generating bonded fragments. For this example, `bonded.inp`, `nonbonded.inp`, and `groups.inp` files are written. Some important information required for the `REAP` mode will be written to the `lssmf.chk` file. Make sure the `lssmf.chk` file is not deleted.

```
# TEST_LABEL: n-heptane
$set_molecule
0 1
C        3.2171710000      0.8842760000      0.2852810000
H        2.5889210000      1.7437160000     -0.0003400000
H        3.2354180000      0.8393510000      1.3882070000
H        4.2428730000      1.1008270000     -0.0552130000
C        2.7017540000     -0.4287940000     -0.3125600000
H        2.6997800000     -0.3550810000     -1.4162340000
H        3.4122590000     -1.2373920000     -0.0652510000
C        1.3020380000     -0.8436370000      0.1668920000
H        1.0873170000     -1.8624800000     -0.2043850000
H        1.3039970000     -0.9190170000      1.2713390000
C        0.1666320000      0.0884510000     -0.2738120000
H        0.3379080000      1.1060480000      0.1216760000
H        0.1838900000      0.1848320000     -1.3764850000
C       -1.2203000000     -0.3892860000      0.1693790000
H       -1.2368210000     -0.4905490000      1.2714170000
H       -1.4015240000     -1.4059910000     -0.2286320000
```

```
C       -2.3613920000      0.5347540000     -0.2691010000
H       -2.1816020000      1.5498180000      0.1311300000
H       -2.3432560000      0.6371130000     -1.3700840000
C       -3.7422770000      0.0473140000      0.1757680000
H       -3.8046130000     -0.0302140000      1.2748300000
H       -4.5400490000      0.7316460000     -0.1555780000
H       -3.9662530000     -0.9510910000     -0.2374760000
$end_molecule

$set_globals
  jobtype energy
  basis cc-pvdz
  aux_basis_corr cc-pvdz-ri
  aux_basis_scf cc-pvdz-jkfit
  method ccsd(t)
  mol_frag true
$end_globals

$set_frag
  software macroqc
  mode sow
  frag_method lssmf
$end_frag

$set_lssmf
  level 3
  nb_level 1
  method_name [scf;mp2;ccd;ccsd;ccsd(t)]
  mode sow
$end_lssmf
```

The `lssmf.chk` file is printed out as below.

```
    Ngroup: 7
    Nnonbfrag: 6
    Nbfrag: 7

    Dimer    Mono1    Mono2
    ========================
    1         1         5
    2         1         6
    3         1         7
    4         2         6
    5         2         7
    6         3         7
```

```
    Coefficients  --->  Label of bonded frags
    ************        **********************
          1         --->     G_1_2_3_4
          1         --->     G_2_3_4_5
          1         --->     G_3_4_5_6
          1         --->     G_4_5_6_7
         -1         --->     G_2_3_4
         -1         --->     G_3_4_5
         -1         --->     G_4_5_6
```

When computations are completed for fragments generated with the `mode sow` option. Then, the same input file should be run with the `mode reap` option, which initiates the reading procedure for each output file and computes the LSSMF energies. When all fragment output files are processed, the following table is printed out. The same table will be prepared for all the methods you specified in `METHOD_NAME` array.

```
    ***************************************************************************
                            CCSD(T) ENERGY
    ***************************************************************************


    Coeff |          Energy |    Coeff*Energy | Bonded Group Name |
    ------|-----------------|-----------------|-------------------|
        1 | -157.9784894936 | -157.9784894936 |         G_1_2_3_4 |
        1 | -157.9782837483 | -157.9782837483 |         G_2_3_4_5 |
        1 | -157.9787063574 | -157.9787063574 |         G_3_4_5_6 |
        1 | -157.9796304517 | -157.9796304517 |         G_4_5_6_7 |
       -1 | -118.7796321247 |  118.7796321247 |           G_2_3_4 |
       -1 | -118.7800829289 |  118.7800829289 |           G_3_4_5 |
       -1 | -118.7800993839 |  118.7800993839 |           G_4_5_6 |
    ---------------------------------------------------------------
     Total bonded energy for CCSD(T) = -275.5752956135
    ---------------------------------------------------------------


Energy of Dimer|Energy of Mono1|Energy of Mono2|Nonbonded I.E.|Nonbonded I.E.|Dimer Names|
---------------|---------------|---------------|--------------|--------------|-----------|
   (hartree)   |     (hartree) |     (hartree) |    (hartree) |   (kcal/mol) |           |
---------------|---------------|---------------|--------------|--------------|-----------|
  -80.77367284 |   -40.38692204 |   -40.38640503 |   -0.00034577 |   -0.21697453 |    G_1_G_5 |
  -80.77342760 |   -40.38692204 |   -40.38642795 |   -0.00007761 |   -0.04870338 |    G_1_G_6 |
  -80.77387251 |   -40.38692204 |   -40.38692804 |   -0.00002244 |   -0.01407833 |    G_1_G_7 |
  -80.77296019 |   -40.38634832 |   -40.38642795 |   -0.00018392 |   -0.11541067 |    G_2_G_6 |
  -80.77331843 |   -40.38634832 |   -40.38692804 |   -0.00004207 |   -0.02639750 |    G_2_G_7 |
  -80.77343675 |   -40.38632403 |   -40.38692804 |   -0.00018468 |   -0.11588552 |    G_3_G_7 |


    Nonbonded interaction energies larger than 1.0 kcal/mol.
    ---------------------------------------------------------------------------
     Sum of nonbonded interaction energies for CCSD(T) ==>        -0.000856481000 (hartree)
                                                                  -0.5 (kcal/mol)
    ---------------------------------------------------------------------------

    ***************************************************************************
     Total energy for CCSD(T) = -275.576152094486
    ***************************************************************************
```

- **Mode Sow-2:**

This is an example of a single-point energy computation at the `LSSMF-ccsd(t)/cc-pvdz` level. In this example, the `PSI4` software is employed instead of `MacroQC`İn this example

`level 3` is used for generating bonded fragments. When an external software is used, the options for the external software should be provided in the `FRAG_EXTERNAL` block. `MacroQC` generates all necessary input files for the allowed external software. The nonbonded inputs (`groups.inp`, `nonbonded.inp`, `dimers.inp`, `monomers.inp`) may be given their own options in a second block, `$SET_EXTERNAL_NB` ... `$END_EXTERNAL_NB`, so that the nonbonded corrections are computed at another level of theory than the bonded fragments (see `NB_METHOD` for why that is allowed); without it the nonbonded inputs carry the `$SET_EXTERNAL` block.

```
$set_molecule
0 1
C         3.2171710000      0.8842760000      0.2852810000
H         2.5889210000      1.7437160000     -0.0003400000
H         3.2354180000      0.8393510000      1.3882070000
H         4.2428730000      1.1008270000     -0.0552130000
C         2.7017540000     -0.4287940000     -0.3125600000
H         2.6997800000     -0.3550810000     -1.4162340000
H         3.4122590000     -1.2373920000     -0.0652510000
C         1.3020380000     -0.8436370000      0.1668920000
H         1.0873170000     -1.8624800000     -0.2043850000
H         1.3039970000     -0.9190170000      1.2713390000
C         0.1666320000      0.0884510000     -0.2738120000
H         0.3379080000      1.1060480000      0.1216760000
H         0.1838900000      0.1848320000     -1.3764850000
C        -1.2203000000     -0.3892860000      0.1693790000
H        -1.2368210000     -0.4905490000      1.2714170000
H        -1.4015240000     -1.4059910000     -0.2286320000
C        -2.3613920000      0.5347540000     -0.2691010000
H        -2.1816020000      1.5498180000      0.1311300000
H        -2.3432560000      0.6371130000     -1.3700840000
C        -3.7422770000      0.0473140000      0.1757680000
H        -3.8046130000     -0.0302140000      1.2748300000
H        -4.5400490000      0.7316460000     -0.1555780000
H        -3.9662530000     -0.9510910000     -0.2374760000
$end_molecule

$set_globals
  jobtype energy
  mol_frag true
$end_globals

$set_frag
  software psi4
  mode sow
  frag_method lssmf
$end_frag
```

```
$set_lssmf
    level 3
    nb_level 1
    method_name [scf;mp2;ccd;ccsd;ccsd(t)]
$end_lssmf

$set_external
set {
    basis cc-pvdz
    df_basis_scf cc-pvdz-jkfit
    df_basis_cc cc-pvdz-ri
    guess sad
    reference rhf
    scf_type df
    freeze_core true
    qc_module occ
    mp2_type df
    mp_type df
    cc_type df
}
$end_external
```

According to the methods you choose in the `METHOD_NAME` option, your `PSI4` input file will be prepared as follows:

```
# Psi4 Input File
memory 5 gb
SET {
BASIS CC-PVDZ
DF_BASIS_SCF CC-PVDZ-JKFIT
DF_BASIS_CC CC-PVDZ-RI
GUESS SAD
REFERENCE RHF
SCF_TYPE DF
FREEZE_CORE TRUE
QC_MODULE OCC
MP2_TYPE DF
MP_TYPE DF
CC_TYPE DF
}
molecule G_1_2_3_4 {
0 1
C      3.2171710000       0.8842760000       0.2852810000  #G1
H      2.5889210000       1.7437160000      -0.0003400000  #G1
H      3.2354180000       0.8393510000       1.3882070000  #G1
H      4.2428730000       1.1008270000      -0.0552130000  #G1
C      2.7017540000      -0.4287940000      -0.3125600000  #G2
H      2.6997800000      -0.3550810000      -1.4162340000  #G2
H      3.4122590000      -1.2373920000      -0.0652510000  #G2
```

```
C      1.3020380000      -0.8436370000       0.1668920000  #G3
H      1.0873170000      -1.8624800000      -0.2043850000  #G3
H      1.3039970000      -0.9190170000       1.2713390000  #G3
C      0.1666320000       0.0884510000      -0.2738120000  #G4
H      0.3379080000       1.1060480000       0.1216760000  #G4
H      0.1838900000       0.1848320000      -1.3764850000  #G4
H     -0.8096951316      -0.2478507039       0.0381711382  #G4-CAP
noreorient
nocom
symmetry c1
}
EG_1_2_3_4=energy('CCSD(T)', molecule=G_1_2_3_4)
E1=get_variable('SCF TOTAL ENERGY')
print_out('==>    SCF Energy of Group G_1_2_3_4 =  %20.12f   \n' % E1)
E2=get_variable('MP2 TOTAL ENERGY')
print_out('==>    MP2 Energy of Group G_1_2_3_4 =  %20.12f   \n' % E2)
E3=get_variable('CCD TOTAL ENERGY')
print_out('==>    CCD Energy of Group G_1_2_3_4 =  %20.12f   \n' % E3)
E4=get_variable('CCSD TOTAL ENERGY')
print_out('==>    CCSD Energy of Group G_1_2_3_4 =  %20.12f   \n' % E4)
E5=get_variable('CCSD(T) TOTAL ENERGY')
print_out('==>    CCSD(T) Energy of Group G_1_2_3_4 =  %20.12f   \n' % E5)
clean()
```

- **Mode Sow-3:**

This is an example of a single-point energy computation at the `LSSMF-mp2/cc-pvdz` level. In this example, `level 3` is selected with the `nb_level 3` option. For this example, `bonded.inp`, `dimers.inp`, and `monomers.inp` files are written. Some important information required for the `REAP` mode will be written to the `lssmf.chk` file. Make sure the `lssmf.chk` file is not deleted.

```
# TEST_LABEL: n-heptane
$set_molecule
0 1
C         3.2171710000       0.8842760000       0.2852810000
H         2.5889210000       1.7437160000      -0.0003400000
H         3.2354180000       0.8393510000       1.3882070000
H         4.2428730000       1.1008270000      -0.0552130000
C         2.7017540000      -0.4287940000      -0.3125600000
H         2.6997800000      -0.3550810000      -1.4162340000
H         3.4122590000      -1.2373920000      -0.0652510000
C         1.3020380000      -0.8436370000       0.1668920000
H         1.0873170000      -1.8624800000      -0.2043850000
H         1.3039970000      -0.9190170000       1.2713390000
C         0.1666320000       0.0884510000      -0.2738120000
H         0.3379080000       1.1060480000       0.1216760000
H         0.1838900000       0.1848320000      -1.3764850000
C        -1.2203000000      -0.3892860000       0.1693790000
```

```
H       -1.2368210000     -0.4905490000      1.2714170000
H       -1.4015240000     -1.4059910000     -0.2286320000
C       -2.3613920000      0.5347540000     -0.2691010000
H       -2.1816020000      1.5498180000      0.1311300000
H       -2.3432560000      0.6371130000     -1.3700840000
C       -3.7422770000      0.0473140000      0.1757680000
H       -3.8046130000     -0.0302140000      1.2748300000
H       -4.5400490000      0.7316460000     -0.1555780000
H       -3.9662530000     -0.9510910000     -0.2374760000
$end_molecule

$set_globals
  jobtype energy
  basis cc-pvdz
  aux_basis_corr cc-pvdz-ri
  aux_basis_scf cc-pvdz-jkfit
  method mp2
  mol_frag true
$end_globals

$set_frag
  software macroqc
  mode sow
  frag_method lssmf
$end_frag

$set_lssmf
  level 3
  nb_level 3
  method_name [scf;mp2]
$end_lssmf
```

The `lssmf.chk` file is printed out as below.

```
    Ngroup: 7
    Nnonbfrag: 5
    Nbfrag: 7

           Dimer            Coefficients
      ***************      ***************
        1 <-> 5_6_7               1
        7 <-> 1_2_3               1
      1_2 <-> 6_7                 1
        1 <-> 6_7                -1
        7 <-> 1_2                -1
```

```
 Coefficients of bonded frags:
 1 --> 1
 2 --> 1
 3 --> 1
 4 --> 1
 5 --> -1
 6 --> -1
 7 --> -1


 Coefficients  --->  Label of bonded frags
 ************        **********************
      1         --->     G_1_2_3_4
      1         --->     G_2_3_4_5
      1         --->     G_3_4_5_6
      1         --->     G_4_5_6_7
     -1         --->     G_2_3_4
     -1         --->     G_3_4_5
     -1         --->     G_4_5_6
```

When computations are completed for fragments generated with the `mode sow` option. Then, the same input file should be run with the `mode reap` option, which initiates the reading procedure for each output file and computes the LSSMF energies. When all fragment output files are processed, the following table is printed out. The same table will be prepared for all the methods you specified in the `METHOD_NAME` array.

```
 ***********************************************************************
                          MP2 ENERGY
 ***********************************************************************


 Coeff |          Energy |    Coeff*Energy | Bonded Group Name |
 ------|-----------------|-----------------|-------------------|
     1 | -157.8967782614 | -157.8967782614 |         G_1_2_3_4 |
     1 | -157.8965730873 | -157.8965730873 |         G_2_3_4_5 |
     1 | -157.8970383126 | -157.8970383126 |         G_3_4_5_6 |
     1 | -157.8978930447 | -157.8978930447 |         G_4_5_6_7 |
    -1 | -118.7158761433 |  118.7158761433 |           G_2_3_4 |
    -1 | -118.7163633621 |  118.7163633621 |           G_3_4_5 |
    -1 | -118.7164024508 |  118.7164024508 |           G_4_5_6 |
 ----------------------------------------------------------------
  Total bonded energy for MP2 = -275.4396407497
 ----------------------------------------------------------------


Energy of Dimer|Energy of Mono1|Energy of Mono2|Nonbonded I.E.|Coeff|I.E.*Coeff|Dimer Names|
---------------|---------------|---------------|--------------|-----|----------|-----------|
     (hartree) |     (hartree) |     (hartree) |    (hartree) |     |(kcal/mol)|           |
---------------|---------------|---------------|--------------|-----|----------|-----------|
 -159.07701523 |   -40.35941955 | -118.71718008 |   -0.00041560 |   1 |     -0.26|G_1_G_5_6_7|
 -159.07620847 |   -40.35942736 | -118.71655498 |   -0.00022613 |   1 |     -0.14|G_7_G_1_2_3|
 -159.07335335 |   -79.53647925 |  -79.53659144 |   -0.00028266 |   1 |     -0.18|G_1_2_G_6_7|
 -119.89610130 |   -40.35941955 |  -79.53659144 |   -0.00009031 |  -1 |      0.06|  G_1_G_6_7|
 -119.89595802 |   -40.35942736 |  -79.53647925 |   -0.00005141 |  -1 |      0.03|  G_7_G_1_2|
-------------------------------------------------------------------------------------------
  Sum of nonbonded interaction energies for MP2 = -0.0007826705 (hartree)
```

```
                                                -0.5          (kcal/mol)
-------------------------------------------------------------------------------------------

 ***************************************************************************
  Total energy for MP2 = -275.440423420194
 ***************************************************************************
```

- **Assigning Charge to Groups:**

This is an example of a single-point energy computation at the `LSSMF-mp2/cc-pvdz` level. In this example, `level 3` is selected with the `nb_level 1` option. `CARBAMINO_CHARGE INPUT_FCHARGE` option must be entered to specify the charge for each group in the input file.

```
# TEST_LABEL: C5H10N2O4
$set_molecule
0 1
O      -9.068110    0.722660   -1.519780 -1
O      -7.011900   -0.029570   -1.195380
C      -8.320340   -0.239900   -1.439050
C      -8.824910   -1.662870   -1.642330
C     -10.350260   -1.707510   -2.035820
C     -10.872490   -3.096450   -2.509370
C     -11.305090   -1.091120   -1.024200
H     -10.401440   -1.065870   -2.945110
O     -10.882560   -0.675020    0.185400
O     -12.482580   -0.951530   -1.316430 -1
N     -11.205560   -4.042800   -1.436050 +1
H     -11.613770   -4.901990   -1.870310
H     -11.924410   -3.639390   -0.794140
H     -10.150260   -3.553080   -3.221100
H     -11.811830   -2.916030   -3.080360
N      -8.472060   -2.490180   -0.477890 +1
H      -8.251990   -2.050510   -2.515240
H      -9.079230   -2.293020    0.347010
H      -7.477340   -2.320770   -0.202850
H     -10.379930   -4.342940   -0.885470
H      -8.505930   -3.498690   -0.730410
$end_molecule

$set_globals
  jobtype energy
  basis cc-pvdz
  aux_basis_corr cc-pvdz-ri
  aux_basis_scf cc-pvdz-jkfit
  method mp2
```

```
  mol_frag true
  memory 5000
$end_globals

$set_frag
  software macroqc
  mode sow
  frag_method lssmf
$end_frag

$set_lssmf
  level 3
  nb_level 1
  method_name [scf;mp2]
  delta_sb 0.40
  delta_db 0.07
  delta_tb 0.25
  carbamino_charge input_fcharge
$end_lssmf
```

The `groups.inp` file will appear as follows.

```
$SET_GLOBALS
MOL_FRAG FALSE
MEMORY 5000
AUX_BASIS_CORR CC-PVDZ-RI
AUX_BASIS_SCF CC-PVDZ-JKFIT
BASIS CC-PVDZ
JOBTYPE ENERGY
METHOD MP2
$END_GLOBALS

$SET_FRAG
  SOFTWARE MACROQC
  MODE SOW
  FRAG_METHOD LSSMF
$END_FRAG

$SET_LSSMF
DELTA_DB 0.07
DELTA_SB 0.4
DELTA_TB 0.25
LEVEL 3
NB_LEVEL 1
CARBAMINO_CHARGE INPUT_FCHARGE
METHOD_NAME [SCF;MP2]
$END_LSSMF


$SET_MOLECULE G1
-1 1
```

```
O         -9.0681100000          0.7226600000         -1.5197800000
O         -7.0119000000         -0.0295700000         -1.1953800000
C         -8.3203400000         -0.2399000000         -1.4390500000
H         -8.6755307237         -1.2415959868         -1.5821484211  #CAP
$END_MOLECULE

@@@

$SET_MOLECULE G2
1 1
N         -8.4720600000         -2.4901800000         -0.4778900000
H         -9.0792300000         -2.2930200000          0.3470100000
H         -7.4773400000         -2.3207700000         -0.2028500000
H         -8.5059300000         -3.4986900000         -0.7304100000
H         -8.7168946939         -1.9161281633         -1.2858687755  #CAP
$END_MOLECULE

@@@

$SET_MOLECULE G3
0 1
C         -8.8249100000         -1.6628700000         -1.6423300000
H         -8.2519900000         -2.0505100000         -2.5152400000
H         -8.4697192763         -0.6611740132         -1.4992315789  #CAP
H         -8.5680736054         -2.2650616327         -0.7947444218  #CAP
H         -9.8986761184         -1.6942942105         -1.9193262500  #CAP
$END_MOLECULE

@@@

$SET_MOLECULE G4
0 1
C        -10.3502600000         -1.7075100000         -2.0358200000
H        -10.4014400000         -1.0658700000         -2.9451100000
H         -9.2764938816         -1.6760857895         -1.7588237500  #CAP
H        -10.7178824342         -2.6852506579         -2.3691742763  #CAP
H        -11.0224100658         -1.2736038816         -1.3236927632  #CAP
$END_MOLECULE

@@@

$SET_MOLECULE G5
0 1
C        -10.8724900000         -3.0964500000         -2.5093700000
H        -10.1502600000         -3.5530800000         -3.2211000000
H        -11.8118300000         -2.9160300000         -3.0803600000
H        -10.5048675658         -2.1187093421         -2.1760157237  #CAP
H        -11.1149287075         -3.7852897959         -1.7281098639  #CAP
$END_MOLECULE

@@@

$SET_MOLECULE G6
-1 1
```

```
C       -11.3050900000          -1.0911200000          -1.0242000000
O       -10.8825600000          -0.6750200000           0.1854000000
O       -12.4825800000          -0.9515300000          -1.3164300000
H       -10.6329399342          -1.5250261184          -1.7363272368  #CAP
$END_MOLECULE

@@@

$SET_MOLECULE G7
1 1
N       -11.2055600000          -4.0428000000          -1.4360500000
H       -11.6137700000          -4.9019900000          -1.8703100000
H       -11.9244100000          -3.6393900000          -0.7941400000
H       -10.3799300000          -4.3429400000          -0.8854700000
H       -10.9744502041          -3.3861489796          -2.1808026531  #CAP
$END_MOLECULE

@@@
```

20.2.2 LSSMF Options

- **NBONDED_CUTOFF_TYPE**
  Type of nonbonded cutoff.
  TYPE: String
  DEFAULT: DCRR
  OPTIONS: DCRR, DBE
  - DCRR: The ratio of distance to covalent radii
  - DBE: Distance-based elimination.

- **METHOD_NAME**
  Which methods do you want to integrate with the LSSMF approach? It works with the `REAP` and `SOW` options. The same string should be used for the `REAP` and `SOW` options. This option accepts multiple method names since energies of several methods can be available in output files. Use ";" character to specify multiple method names, such as `[SCF;MP2;CCD]`. This option is used directly for `MacroQC` and Psi4. For Q-Chem and ORCA, see the `ENERGY_LINE` option.
  TYPE: String Array
  DEFAULT: [SCF]
  OPTIONS: User defined method(s) name.

- **ENERGY_LINE**
  The string before energy values can be specified here so that `MacroQC` can grep the energy from outputs of external software. It works with the `REAP` option and is mainly used with Q-Chem and ORCA.
  TYPE: LINE
  DEFAULT: "Total Energy"
  OPTIONS: User defined energy line.

- **CARBAMINO_CHARGE**
  How the formal charges of the groups are found.
  TYPE: STRING
  DEFAULT: AUTO
  OPTIONS: AUTO , INPUT_FCHARGE

  - AUTO : The charges are found from the structure. With `FUNCTIONAL_GROUPS TRUE` (the default) every charged group of the functional-group library — carboxylate, alkoxide, thiolate, sulfonate, hydrogen sulfate and the phosphate anions (-1, the phosphate dianion -2), ammonium, amidinium, guanidinium, imidazolium and pyridinium ($+1$) — carries the charge given in the library. With the library off, three rules apply: a carbon with three heavy neighbours, two of them oxygens bonded to nothing else, is a carboxylate (-1); a capped group of 11 electrons is an ammonium ($\mathtt{R\text{-}NH_3^+}$) and one of 33 electrons a guanidinium ($+1$). In both cases the group charges are summed and compared with the charge of the molecule, and a mismatch is printed as a warning: a charged group that neither the library nor the rules know must be given with `INPUT_FCHARGE`.
  - INPUT_FCHARGE : The charge of each group is read from the input file, written after the coordinates of one atom of the group in `$set_molecule` (e.g. `O 1.23 4.56 7.89 -1`).

- **RING_MAX_MERGE**
  The largest ring, in heavy atoms, whose atoms are kept together in one group. A ring cannot be cut in one place: a fragment that leaves part of a ring out has cut it twice, and its two cap hydrogens then stand in for atoms of the same missing arc. Fused rings follow on their own (every bond of decalin lies on a six-ring). A value below 3 switches the merge off.
  TYPE: Integer
  DEFAULT: 6
  OPTIONS: n User-defined value.

- **FUNCTIONAL_GROUPS**
  Keep every functional group in one group. A fragmentation cuts single bonds, and which single bonds may be cut is a chemical question that the distance thresholds (`DELTA_DB`) answer by accident: the C–O of a carboxylic acid (1.34–1.36 Å), the C–O of an ester (1.33–1.36 Å) and the C–N of an amide (1.33–1.38 Å) all lie within a few hundredths of an angstrom of the double-bond threshold, the C–O of an acetal (1.38–1.43 Å), the S–S of a disulfide (2.04 Å) and the O–O of a peroxide (1.47 Å) lie beyond it, and cutting any of them puts a cap hydrogen in the middle of a resonance that no level of the expansion repairs. With this option the functional groups are recognised on the covalent graph — element, number of bonds, number of hydrogens, ring membership, never the bond order — from the library `share/macroqc/fragment/functional_groups.dat`, and the atoms of each group are put in one group. The library names 71 groups (carboxylic acids and carboxylates, esters, amides, anhydrides, carbamates, ureas, imines, oximes, amidines, guanidines, nitro, nitriles, azides, alcohols, phenols,

ethers, acetals, peroxides, thiols, disulfides, sulfoxides, sulfones, sulfonic acids and sulfonamides, sulfates, phosphates, phosphonates, boronic acids, ...) together with each group's formal charge and a reference geometry: the experimental gas-phase structure of the smallest molecule containing the group where one has been measured, and otherwise an MP2/cc-pVDZ structure. The matched atoms are compared with that geometry (see `FG_GEOM_CHECK`), and under `CARBAMINO_CHARGE AUTO` the charge of a charged group (carboxylate, ammonium, guanidinium, imidazolium, sulfonate, phosphate anions, ...) is taken from the library, in place of the electron-count rules that only knew three groups. The output lists every group found, its atoms, its charge and its largest deviation from the reference bond lengths and angles, and warns when the group charges do not add up to the charge of the molecule. The alkyl carbon a group hangs on is not part of it — the sigma bond to it stays cuttable, as in SMF. Set it to `FALSE` to fragment by the distance thresholds alone, as before.
TYPE: Boolean
DEFAULT: TRUE
OPTIONS: TRUE, FALSE

- **FG_LIBRARY**
  The functional-group library to read. `DEFAULT` is `$MQCDATADIR/fragment/functional_groups.dat`; any other value is a path to a file in the same format, which replaces it. The format is documented at the top of the default file: each entry carries a SMARTS-like pattern (element, `Xn` total bonds, `Hn` hydrogens, `Dn` heavy neighbours, `R` on a held ring, alternatives such as `[Cl,Br,I]`, branches and ring closures; bond symbols mean "bonded" and the bond order is not tested), the pattern atoms that are context only, the group's charge, its model molecule, the source of its geometry and the reference bond lengths and angles by pattern atom index.
  TYPE: LINE
  DEFAULT: DEFAULT
  OPTIONS: DEFAULT, or a path.

- **FG_LIBRARY_EXTRA**
  A second library file read after `FG_LIBRARY`, for a user's own groups.
  TYPE: LINE
  DEFAULT: NONE
  OPTIONS: NONE, or a path.

- **FG_CONJUGATION**
  Hold every single bond between two π atoms in one group. The double-bond rule keeps C=C and C=O whole but says nothing about the single bond joining two of them: the central bond of butadiene (1.48 Å), the ring-to-ring bond of biphenyl (1.49 Å) and the ring-to-carboxyl bond of benzoic acid (1.48 Å) are all single by the thresholds and all conjugated. A π atom is one that carries a multiple bond to a heavy atom other than a halogen, or a three-coordinate carbon or nitrogen whose bond angles sum to a plane (an amide nitrogen; a pyramidal amine is not one). A fully conjugated molecule becomes a single group with this on.

TYPE: Boolean
DEFAULT: TRUE
OPTIONS: TRUE, FALSE

- **FG_GEOM_CHECK**
  What to do with a functional group whose bond lengths or angles deviate from the library's reference geometry by more than `FG_BOND_TOL` or `FG_ANGLE_TOL`. `WARN` reports it and holds the group whole anyway; `STRICT` does not merge it, on the first geometry of the run only (once the topology is held by `FREEZE_TOPOLOGY`, the groups are held with it); `NONE` does not compare.
  TYPE: String
  DEFAULT: WARN
  OPTIONS: WARN, STRICT, NONE

- **FG_BOND_TOL**
  Tolerance, in Å, on the reference bond lengths of a functional group.
  TYPE: Double
  DEFAULT: 0.15
  OPTIONS: n User-defined value.

- **FG_ANGLE_TOL**
  Tolerance, in degrees, on the reference bond angles of a functional group.
  TYPE: Double
  DEFAULT: 15.0
  OPTIONS: n User-defined value.

- **NPIECE_B**
  Specifies the number of ASCII files that the bonded fragments file will be split. Used with `SOW` or `SPLIT` options. The same value should be used for the `SOW` and `REAP` options.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **NPIECE_NB**
  Specifies the number of ASCII files that the nonbonded fragments file will be split. Used with `SOW` or `SPLIT` options. The same value should be used for the `SOW` and `REAP` options.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **NPIECE_G**
  Specifies the number of ASCII files that the groups' file will be split. Used with `SOW` or `SPLIT` options. The same value should be used for the `SOW` and `REAP` options.
  TYPE: Integer

DEFAULT: 0
OPTIONS: n User-defined value.

- **NPIECE_D**
  Specifies the number of ASCII files that the `dimers.inp` file will be split. Used with `SOW` or `SPLIT` options, typically with `nb_level 2` or `nb_level 3`.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **NPIECE_M**
  Specifies the number of ASCII files that the `monomers.inp` file will be split. Used with `SOW` or `SPLIT` options, typically with `nb_level 2` or `nb_level 3`.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **LEVEL**
  Fragmentation level for bonded fragments. The main fragments contain `LEVEL`+1 groups, so a molecule must have at least `LEVEL`+1 groups to be fragmented at all.

  Levels 1 and 2 have been withdrawn. Level 3 is the lowest at which the method is accurate, which is what the paper introducing LSSMF concluded as well. Level 2 is an order of magnitude worse — 6.0 kcal/mol on n-heptane and 15.8 on 2,2,3-trimethylbutane, against 0.01 and 0.12 at level 3. Level 1 is not merely inaccurate but ill-defined: its main fragments are bonded pairs, so two groups separated by a single third group are never together in a bonded fragment and fall through to the nonbonded step, where their two cap hydrogens both stand in for the same missing atom and land 0.7 Å apart. The resulting error is hundreds of kcal/mol.
  TYPE: Integer
  DEFAULT: 3
  OPTIONS: 3, 4, 5, 6, 7, 8, 9

- **NB_LEVEL**
  Fragmentation level for nonbonded fragments.
  TYPE: Integer
  DEFAULT: 1
  OPTIONS: 1, 2, 3

  - 1 : Conventional group-based nonbonded treatment using `nonbonded.inp` and `groups.inp`.
  - 2 : Dimer/monomer based nonbonded treatment using `dimers.inp` and `monomers.inp`.
  - 3 : Higher-order dimer/monomer based nonbonded treatment. If level 3 nonbonded fragments cannot be formed for a given case, LSSMF automatically continues with `nb_level 2`.

- **NB_METHOD**
  The method of the nonbonded fragments — the groups, the pair dimers, the counterpoise and embedded monomers, the dimers and monomers of NB_LEVEL 2 and 3, and the CFM dimers. The bonded fragments are computed with the METHOD of the GLOBALS block; with this option the nonbonded fragments are computed with another one. A nonbonded term is an interaction energy, $E_{ij} - E_i - E_j$, whose three pieces have to agree with each other but not with the bonded expansion: the bonded energy is a sum of total energies whose coefficients add to one and fixes the level of theory of the result, while the nonbonded sum is a correction with coefficients that add to zero, and may be taken at any level that describes the interaction. So an LSSMF-CCSD(T) backbone can be combined with MP2 nonbonded corrections, or a DFT backbone with an MP2 dispersion correction, at a fraction of the cost. The same holds term by term for the gradient, which assembles the bonded and nonbonded gradients separately, and for an excitation energy, where the nonbonded piece is $\Delta E_{\text{pair}} - \Delta E_{\text{excited monomer}}$. In MODE RUN the option is applied once per phase, so it reaches the forked workers of FRAG_NPROC and the MPI ranks; in MODE SOW the option block of groups.inp, nonbonded.inp, dimers.inp and monomers.inp carries the nonbonded level, and REAP reads it back with NB_METHOD_NAME. The output states both levels under "Level of theory" and in the energy table. For an external program, the same is done with a $SET_EXTERNAL_NB block (see SOFTWARE). NONE means the GLOBALS method.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, or any value METHOD accepts.

- **NB_BASIS**
  The orbital basis of the nonbonded fragments; NONE means the GLOBALS basis. May be combined with NB_METHOD or used on its own, e.g. a larger basis for the dispersion-bound pairs than for the covalent backbone.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, or a basis-set name.

- **NB_AUX_BASIS_SCF**
  The SCF (JK) auxiliary basis of the nonbonded fragments; NONE means AUX_BASIS_SCF of the GLOBALS block.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, or an auxiliary basis-set name.

- **NB_AUX_BASIS_CORR**
  The correlation (RI) auxiliary basis of the nonbonded fragments; NONE means AUX_BASIS_CORR of the GLOBALS block.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, or an auxiliary basis-set name.

- **NB_REFERENCE**
  The reference wavefunction of the nonbonded fragments; `NONE` means `REFERENCE` of the `GLOBALS` block. Needed when the two levels differ in kind, e.g. a Kohn–Sham backbone (`RKS`) with wavefunction nonbonded corrections (`RHF`).
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, RHF, UHF, ROHF, RKS, UKS

- **NB_DFT_FUNCTIONAL**
  The density functional of the nonbonded fragments; `SAME` means `DFT_FUNCTIONAL` of the `SCF` block, and `NONE` means no functional (a Hartree–Fock or post-HF nonbonded level under a DFT backbone, together with `NB_REFERENCE RHF`).
  TYPE: String
  DEFAULT: SAME
  OPTIONS: SAME, or any value `DFT_FUNCTIONAL` accepts.

- **NB_METHOD_NAME**
  `MODE REAP`: the method whose energy line is read from the nonbonded outputs (`groups.out`, `nonbonded.out`, `dimers.out`, `monomers.out`) when they were sown at another level than the bonded ones. One entry serves every `METHOD_NAME` entry; a list is read in step with `METHOD_NAME`. `NONE` reads the `METHOD_NAME` line from the nonbonded outputs too.
  TYPE: String array
  DEFAULT: [NONE]
  OPTIONS: e.g. `[SCF]`, `[MP2;MP2]`

- **NB_ENERGY_LINE**
  `MODE REAP`: the same thing as `NB_METHOD_NAME` for the readers that search for `ENERGY_LINE` (the frequency combine, and the `NB_LEVEL 2` and `3` monomers of ORCA outputs). `NONE` means `ENERGY_LINE`.
  TYPE: Line
  DEFAULT: NONE
  OPTIONS: A quoted line of text.

- **NCIE_PRINT_TOL**
  Threshold value (in kcal mol^{-1}) to print noncovalent interaction energies to the output file.
  TYPE: Integer
  DEFAULT: 1.0
  OPTIONS: n User-defined value.

- **NB_CUTOFF_MAX**
  Tolerance value for the DCRR and DBE algorithms. If the distance between the closest atoms of two groups is larger than `NB_CUTOFF_MAX`, then this nonbonded fragment is disregarded. TYPE: Integer
  DEFAULT: 5.0
  OPTIONS: n User-defined value.

- **EC_MAX_DISTANCE**
  Maximum distance (in Å) used for external-charge embedding domains. An atom outside a fragment carries a point charge if some atom of that fragment is closer than this. The charge is faded out over the last `EC_SWITCH_WIDTH` Å rather than cut, so that the energy stays continuous when an atom crosses the boundary. This option is active when `EMBEDDING_CHARGE` is not `NONE`.
  TYPE: Double
  DEFAULT: 10.0
  OPTIONS: n User-defined value.

- **RING_MAX_MERGE**
  The largest ring, counted in heavy atoms, whose atoms are kept together in a single group instead of being distributed over several. Set to 0 to disable the merge and split rings as before.

  A chain can be cut anywhere: one bond, one cap, and the two halves are separate molecules. A ring cannot. Any fragment that leaves part of a ring out has cut it in two places, and the two caps then stand in for atoms of the same missing arc; when that arc is a single group they stand in for the same atom, and the two cap hydrogens land about 0.7 Å apart. The energy pays for it even where the caps do not collide — cyclohexane at LSSMF(3,1) is 6.7 kcal/mol from the exact answer where n-hexane, the same size and the same number of groups, is 0.01, and its expansion contains fragments made of two groups on opposite sides of the ring that are not bonded to each other at all. On butylcyclohexane the merge takes the error from −6.49 to −0.02 kcal/mol.

  Half of this already happened without the option: an aromatic C–C bond is 1.39 Å, below the double-bond threshold `DELTA_DB`, so benzene was always one group. Cyclohexane's bonds are 1.53 Å and it came out as six. The default puts the two on the same footing.

  The test is applied per bond: a bond lies on a small ring when its two atoms are joined by some other path of at most `RING_MAX_MERGE`−1 bonds. Fused systems follow on their own, since every bond of a fused pair lies on a small ring, so decalin becomes one group; a macrocycle with a six-ring fused to it keeps its macrocyclic bonds splittable. Rings of seven heavy atoms and up cost less than 1 kcal/mol at level 3 and are left alone by the default.

  Note that merging can leave a molecule with fewer than `LEVEL`+1 groups, in which case it is not fragmented at all and is computed exactly. Cyclohexane and pyrrolidine are such molecules.
  TYPE: Integer
  DEFAULT: 6
  OPTIONS: n User-defined value; 0 disables the merge.

- **DELTA_SB**
  Tolerance value (in Å) to identify an inter-atomic distance as a single bond.
  TYPE: Integer

DEFAULT: 0.40
OPTIONS: n User-defined value.

- **DELTA_DB**
  Tolerance value (in Å) to identify an inter-atomic distance as a double bond.
  TYPE: Integer
  DEFAULT: 0.05
  OPTIONS: n User-defined value.

- **DELTA_TB**
  Tolerance value (in Å) to identify an inter-atomic distance as a triple bond.
  TYPE: Integer
  DEFAULT: 0.25
  OPTIONS: n User-defined value.

- **BSSE_TYPE**
  Type of basis-set-superposition-error treatment for nonbonded fragments. With `CP`, each monomer of a nonbonded pair is recomputed in the basis of the pair — its partner's atoms and caps become ghosts — so that the correction $E_{\mathrm{dimer}} - E_1 - E_2$ no longer carries the superposition error. Available in `MODE SERIAL` as well as `SOW`, for `NB_LEVEL 1` and `2` with `CFM` off, and for gradients as well as energies. At `NB_LEVEL 2` the two sides of every three-body term $G_a \cdots G_b G_c$ and of every renormalization term are each recomputed in the basis of that term, with the other side ghosted, and named for it (`G_a_GH_b_c`, `G_b_c_GH_a`), as `MODE SOW` writes them to `monomers.inp`. `NB_LEVEL 3` has no counterpoise in either mode. On a cc-pVDZ water dimer it removes 2.15 kcal/mol.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, CP

- **READ_BONDED**
  Combine bonded fragment energies in `REAP` mode.
  TYPE: Bool
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **READ_NONBONDED**
  Combine nonbonded fragment energies in `REAP` mode.
  TYPE: Bool
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **READ_GROUP**
  Combine group/monomer energies in `REAP` mode.
  TYPE: Bool
  DEFAULT: TRUE
  OPTIONS: TRUE, FALSE

- **CFM**
  Activates the CFM-style nonbonded fragment construction used together with the LSSMF machinery.
  TYPE: Bool
  DEFAULT: FALSE
  OPTIONS: TRUE, FALSE

- **EMBEDDING_CHARGE**
  Whether each fragment is computed in the field of point charges standing on the atoms around it, and where those charges come from.

  `EEQ` takes them from charge equilibration (the EEQ2019 model, the one the D4 dispersion correction uses). It needs no calculation at all: its parameters are indexed by the atomic number and the chemistry enters through a smooth coordination number. That smoothness is why it is the only source offered — the charges are an explicit analytic function of the nuclear coordinates, so an embedded run can have an analytic gradient. A population analysis cannot: differentiating it would need the coupled-perturbed response of every fragment density.

  Charges are placed on every atom within `EC_MAX_DISTANCE` of the fragment that is not itself in the fragment, subject to `EC_LINK_SCHEME`. The mutual repulsion of those charges is not included in the fragment energy (see `EC_EC_REPULSION` in the `GLOBALS` block).
  The monomers of every nonbonded term are computed in their own charge field, the partner's charges included: the bonded expansion counts the electrostatics of each pair outside its window twice through the charges, once as each group's density in the other's charge, and only a monomer that sees its partner's charge takes that back (the rule of the electrostatically embedded pairwise-additive expansion). The counterpoise monomers are embedded likewise, their ghost atoms carrying their charges. Pairs no nonbonded term contains are handled by `EC_FAR_CORRECTION`.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, EEQ

- **EC_FAR_CORRECTION**
  Take back, once and classically, the electrostatics of the group pairs that the bonded expansion counts twice through the embedding charges but that no nonbonded term corrects: the pairs beyond `NB_CUTOFF_MAX` and inside `EC_MAX_DISTANCE`. For every such pair (i, j) outside the bonded window the term $-w_{ij} \sum_{a\in i} \sum_{b\in j} S_{ab}\, q_a q_b / r_{ab}$ is added, with w_{ij} one minus the sum of the coefficients of the bonded fragments holding both groups, q the model charges and S_{ab} the mean of the switching weights with which b's charge entered the fragments of i and a's the fragments of j, taken over the two groups' own atoms. The gradient carries the explicit derivative and, through the charges, the EEQ response. In `MODE SOW` the value is written to `lssmf.chk` and `REAP` adds it. The output reports how many pairs are inside the window, corrected, and far. The classical model represents a pair's electrostatics only at a distance, which is what a far pair is at the default cutoffs; with a short `NB_CUTOFF_MAX` the "far" pairs are not

far and the point charges misrepresent them, so the cutoff should not be lowered below its default when embedding charges are used.
TYPE: Boolean
DEFAULT: TRUE
OPTIONS: TRUE, FALSE

- **EC_LINK_SCHEME**
  What to do with the charge of the atom a cap hydrogen stands in for. A cap replaces the atom across a cut bond; that atom is not in the fragment's groups, so without this it is also given a point charge — half an angstrom from the cap that already represents it, which is a large artefact.
  `SHIFT` spreads its charge equally over its bonded neighbours outside the fragment, so the environment keeps its total charge. `DELETE` drops it. Which atoms these are is decided from the bond topology alone, never from `EC_MAX_DISTANCE`: a redistribution that depended on the cutoff would move a whole charge the moment a neighbour crossed it, and no switching function applied afterwards could smooth that. `NONE` leaves it in place, which is not correct and is retained only to reproduce results from before this option existed.
  TYPE: String
  DEFAULT: SHIFT
  OPTIONS: SHIFT, DELETE, NONE

- **EC_SWITCH_WIDTH**
  Width (in Å) of the window over which an embedding charge is switched off as it leaves `EC_MAX_DISTANCE`. A charge is at full strength while some atom of the fragment is closer than `EC_MAX_DISTANCE` − `EC_SWITCH_WIDTH`, and fades smoothly to zero as the last of them reaches the cutoff.

  Without it the cutoff is a step: an atom crossing the boundary makes the fragment energy jump — up to 8×10^{-4} hartree on propane — and a step has no derivative, so the analytic gradient is not valid there. That is the region a geometry optimization is most likely to visit, since a line search that moves an atom across a boundary then reports an energy rise that no reduction of the step size will remove. The switch enters the gradient too, through the dependence of the charge sizes on the nuclear positions.

  A narrow window keeps the charge field closest to the unswitched one but makes the switching force sharp; a wide one is gentler but attenuates charges well inside the cutoff. The default fades over the last couple of Å, where the charges contribute least. Setting it to zero restores the hard cutoff, and is meant only for reproducing results from before this option existed.
  TYPE: Double
  DEFAULT: 2.0
  OPTIONS: n User-defined value.

- **CHARGE_METHOD**
  Electronic structure method used to generate embedding charges when an *ab initio* charge model is requested. Not used by `EMBEDDING_CHARGE EEQ`, which computes no

wavefunction.
TYPE: String
DEFAULT: SCF
OPTIONS: User-defined method name.

- **CHARGE_BASIS**
  Common basis specification for the charge computation. If this option is used, it overrides the need to set separate SCF and correlation auxiliary charge bases.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: User-defined basis name.

- **CHARGE_BASIS_SCF**
  SCF auxiliary basis specification for embedding-charge computations.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: User-defined basis name.

- **CHARGE_BASIS_CORR**
  Correlation auxiliary basis specification for embedding-charge computations.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: User-defined basis name.

20.3 Generalized N-Body Fragmentation

N-body fragmentation is a generalized function for computing interaction energy [76]. The current implementation supports MacroQC and ORCA workflows through the common `FRAG` block options `SOFTWARE` and `MODE`. The allowed modes are `SERIAL`, `SOW`, `REAP`, and `SPLIT`.

The subset energy contributions are defined by the following equation 40.

$$\varepsilon^{(k)}_{i_1 i_2 i_3 \ldots i_k} = E^{(k)}_{i_1 i_2 i_3 \ldots i_k} - \Bigg[\sum_{\substack{j_1 \\ j_1 \in \{I^k_l\}}} \varepsilon^{(1)}_{j_1} + \sum_{\substack{j_1 < j_2 \\ j_1, j_2 \in \{I^k_l\}}} \varepsilon^{(2)}_{j_1 j_2} + \cdots + \sum_{\substack{j_1 < j_2 < \ldots j_{k-1} \\ j_1, j_2, \ldots, j_{k-1} \in \{I^k_l\}}} \varepsilon^{(k-1)}_{j_1 j_2 \ldots j_{k-1}} \Bigg] \quad (40)$$

The total energy of a system in the N-body fragmentation can be expressed as

$$E_{tot} = \sum_{i=1}^{N} E_{i_1} + \sum_{i_1 < i_2} \varepsilon^{(2)}_{i_1 i_2} + \sum_{i_1 < i_2 < i_3} \varepsilon^{(3)}_{i_1 i_2 i_3} + \cdots + \sum_{i_1 < i_2 < i_3 \ldots N} \varepsilon^{(N)}_{i_1 i_2 i_3 \ldots N} \quad (41)$$

20.3.1 Input Examples

*Counterpoise corrected SCF energies for the H2O trimer:

```
$set_body
O     -1.462504   -0.658178   -0.081729
H     -0.594572   -1.091311   -0.027749
H     -2.015066   -1.104422    0.561406
--
O      1.306669   -0.928376   -0.090452
H      1.991655   -1.171733    0.533655
H      1.220274    0.037563   -0.020863
--
O      0.154138    1.592165    0.107468
H     -0.641643    1.036240    0.054523
H      0.052929    2.248781   -0.583270
$end_body

$set_globals
  basis cc-pvdz
  aux_basis_scf cc-pvdz-jkfit
  aux_basis_corr cc-pvdz-ri
  method scf
  reference rhf
  jobtype energy
  nocom true
  memory 5000
$end_globals

$set_frag
  software macroqc
  mode run
  frag_method nbody
$end_frag

$set_nbody
  max_nbody 3
  embedding_charge false
  charge_method scf
  charge_basis none
  charge_type mulliken
  bsse_type cp
$end_nbody
```

In the above example, `max_nbody 3` requests up to three-body contributions, while `bsse_type cp` activates the counterpoise-corrected N-body expansion. If `mode sow` is selected, `MacroQC` writes all subsystem inputs to `nbody.inp`. When fragment computations are completed, the same job can be executed with `mode reap` to collect subsystem energies and to print the 1-body, 2-body, ..., `max_nbody`-body interaction contributions.

20.3.2 N-BODY Options

- **MAX_NBODY**
  Number of maximum n-body to compute.
  TYPE: Integer
  DEFAULT: 2
  OPTIONS: n User-defined value.

- **NPIECE**
  Specifies the number of ASCII files that the nbody file will be split. Used with `SOW` or `SPLIT` options. The same value should be used for the `SOW` and `REAP` options.
  TYPE: Integer
  DEFAULT: 0
  OPTIONS: n User-defined value.

- **EMBEDDING_CHARGE**
  Requests embedding charges for atoms that are outside of the active N-body fragment when `BSSE_TYPE` is not `CP`.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: FALSE, TRUE.

- **CHARGE_METHOD**
  Method to compute embedding charges for monomers.
  TYPE: String
  DEFAULT: SCF
  OPTIONS: User defined available method name.

- **CHARGE_BASIS**
  Basis set to be used in embedding charge computing.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: User defined basis set. If no selection is made, the `GLOBALS` basis selection is taken into account.

- **CHARGE_TYPE**
  Type of charge to compute.
  TYPE: String
  DEFAULT: MULLIKEN
  OPTIONS: MULLIKEN, LOWDIN.

- **BSSE_TYPE**
  Type of BSSE correction to compute.
  TYPE: String
  DEFAULT: NOCP
  OPTIONS: NOCP, CP.

- NOCP: Plain molecular interaction energy.
- CP: Counterpoise correction. In this case, ghost atoms are written for the inactive bodies instead of embedding charges.

- **ENERGY_LINE**
  The string before energy values can be specified here so that `MacroQC` can grep the energy from outputs of external software. It works with the `REAP` option and is mainly used for ORCA.
  TYPE: LINE
  DEFAULT: "Total Energy"
  OPTIONS: User defined energy line.

20.4 Molecular Fractionation with Conjugate Caps (MFCC)

The molecular fractionation with conjugate caps (MFCC) approach is available in `MacroQC` for residue-based protein–ligand interaction energy computations.[77] In the current implementation, the protein is decomposed into capped fragments and conjugate-cap fragments, and the ligand is treated as a separate subsystem. The resulting interaction energy is assembled from the fragment, conjugate-cap, and ligand energies.

Let F_k denote the kth capped fragment, CC_k the corresponding conjugate-cap fragment, and L the ligand. The MFCC interaction energy is computed as

$$E_{\mathrm{int}}^{\mathrm{MFCC}} = \sum_{k=1}^{N-2} E(F_k - -L) - \sum_{k=1}^{N-2} E(F_k) - \sum_{k=1}^{N-3} E(CC_k - -L) + \sum_{k=1}^{N-3} E(CC_k) - E(L) \quad (42)$$

This expression removes the artificial contributions introduced by the capping procedure while retaining the local protein environment around the ligand.

The current MFCC workflow supports the common `FRAG` block options `MODE` and `SOFTWARE`. The allowed modes are `SERIAL`, `SOW`, `REAP`, and `SPLIT`. At present, MFCC is implemented for the MacroQC software only. Both Cartesian input with residue/ligand mono labels and PDB-based input can be used. In Cartesian input, ligand atoms should be labeled with mono names beginning with `M`; in PDB input, ligand atoms are read from `HETATM` records.

20.4.1 Input Examples

- **SOW and REAP Modes:**

The following example illustrates an MFCC setup for a residue-based protein–ligand interaction energy computation. In this test system, the protein part is given with residue labels `A1`, `A2`, ..., `A5`, and the ligand is given with the label `M1`.

```
$set_molecule
 0 1
N      -8.98856100     1.31093500    -0.10750100 A1
```

```
C      -7.58029000    1.70272900    0.28886800 A1
C      -6.67699000    0.49314200   -0.07802900 A1
O      -7.25278600   -0.46172700   -0.59631800 A1
H      -7.24763900    2.62605000   -0.24416100 A1
H      -7.51243000    1.93535800    1.37951800 A1
H      -8.95736300    0.33060700   -0.54845200 A1
H      -9.38108800    1.96677700   -0.79913800 A1
H      -9.61328900    1.27682400    0.71211600 A1
N      -5.35116200    0.60500300    0.17832200 A2
C      -4.41898600   -0.52709100   -0.07168800 A2
C      -3.90592100   -0.49891900   -1.54174600 A2
O      -2.94025500   -1.52999100   -1.67389200 A2
C      -3.24339600   -0.33981300    0.91362700 A2
O      -3.16174600    0.68142300    1.58129500 A2
H      -4.93853500   -1.51085100    0.10716300 A2
H      -4.92772600    1.41552900    0.67017300 A2
H      -4.72385200   -0.78117300   -2.23607700 A2
H      -3.48992000    0.48435200   -1.81901700 A2
H      -2.00165200   -1.14509400   -1.65964700 A2
N      -2.37920700   -1.42194700    1.01678400 A3
C      -1.02594800   -1.18707600    1.55570600 A3
C      -0.09241900   -0.72244800    0.42701100 A3
O      -0.49872500   -0.54978900   -0.72031700 A3
H      -1.06448200   -0.42036500    2.37298100 A3
H      -0.65497600   -2.13190700    2.02198300 A3
H      -2.42831000   -2.15729700    0.28629500 A3
N       1.21181000   -0.47443600    0.79090700 A4
C       2.23423700   -0.12794700   -0.22172100 A4
C       2.52682200    1.39914900   -0.18923500 A4
O       3.52116200    1.68603200   -1.16067600 A4
C       3.51880800   -0.90859900    0.12734200 A4
O       3.64114300   -1.47539400    1.20247200 A4
H       1.63608400    1.96564000   -0.52468100 A4
H       2.83665400    1.73938000    0.81284300 A4
H       1.86572700   -0.40031000   -1.25217300 A4
H       4.45384800    1.64719100   -0.74885500 A4
H       1.58496600   -0.71581800    1.71894500 A4
N       4.46568000   -0.95573100   -0.89223300 A5
C       5.86422100   -1.26584200   -0.54012700 A5
C       6.66811000    0.00022900   -0.21175800 A5
O       6.10862900    1.13564600   -0.21107700 A5
O       7.88863800   -0.14475400    0.04682900 A5
H       5.89895700   -1.96728200    0.33454400 A5
H       6.34142800   -1.81654700   -1.38517900 A5
H       4.36351400   -0.29465900   -1.67941600 A5
```

```
Na     8.50838100    1.97001600    0.46184200 1.0 M1
$end_molecule

$set_globals
  basis 6-31g
  aux_basis_corr 6-31g
  aux_basis_scf  6-31g
  method scf
  jobtype energy
  mol_frag true
  nocom true
  memory 1000
$end_globals

$set_scf
  e_convergence 1.0e-8
  d_convergence 1.0e-6
  guess gwh
$end_scf

$set_frag
  software macroqc
  mode sow
  frag_method mfcc
$end_frag

$set_mfcc
  npiece 0
  method_name [SCF]
$end_mfcc
```

In this example, `mol_frag true` requests an MFCC computation. The residue labels `A1`–`A5` define the protein residues, while the `M1` label identifies the ligand. With `mode sow`, `MacroQC` writes the MFCC input files `capped_frags.inp`, `concaps.inp`, `capped_frags_ligand.inp`, `concap_ligand.inp`, and `ligand.inp`. If `npiece` is greater than one, the main fragment files are additionally split into `part` files. Once all fragment jobs are completed, the same input can be executed with `mode reap` by changing only the `mode` keyword in the `frag` block from `sow` to `reap`. The `METHOD_NAME` array is used in the `REAP` step to identify the energy lines that will be collected from the output files.

For this test system, the `REAP` output begins with the following summary:

```
   ============================================
    MFCC residue-based decomposition summary
   ============================================
    Number of protein residues :        5
    Number of ligand atoms     :        1
    Number of capped fragments :        3
    Number of concaps          :        2
```

```
    ==========================================
```

When all fragment output files are processed, the following MFCC energy table is printed:

```
     ************************************************************************
                                  MFCC ENERGY
     ************************************************************************

     Fragment Name |            Energy |     Fragment Type |
     --------------|-------------------|-------------------|
               F_1 |   -624.7409691617 |          FRAGMENT |
               F_2 |   -529.9363159519 |          FRAGMENT |
               F_3 |   -717.2169291655 |          FRAGMENT |
             F_1-L |   -785.2765831286 |   FRAGMENT-LIGAND |
             F_2-L |   -690.5002546627 |   FRAGMENT-LIGAND |
             F_3-L |   -878.0080280581 |   FRAGMENT-LIGAND |
              CC_1 |   -208.4700063606 |             CONCAP |
              CC_2 |   -322.5830114578 |             CONCAP |
            CC_1-L |   -369.0409235471 |     CONCAP-LIGAND |
            CC_2-L |   -483.1477230883 |     CONCAP-LIGAND |
                 L |   -160.5743754032 |             LIGAND |
     ------------------------------------------------------------------------
      Total MFCC interaction energy  =      -0.180647350026 a.u
      Total MFCC interaction energy  =              -113.36 kcal/mol
     ------------------------------------------------------------------------
```

- **Run Mode:**

The same system can also be computed directly with `mode run`:

```
$set_frag
  software macroqc
  mode run
  frag_method mfcc
$end_frag

$set_mfcc
  npiece 0
  method_name [SCF]
$end_mfcc
```

In serial mode, all capped fragments, conjugate-cap fragments, fragment–ligand combinations, and the ligand itself are generated and computed on the fly. Before each subsystem computation, `MacroQC` prints a progress block such as

```
    ********************************************************************
                         Computing F_1
    ********************************************************************
```

This makes it straightforward to identify which subsystem is currently being computed. After all fragment computations are completed, the same MFCC energy summary table is printed in the main output file.

20.4.2 MFCC Options

- **NPIECE**
  Specifies the number of ASCII files that the MFCC sow files will be split into. Used

with SOW, REAP, or SPLIT options. The same value should be used for the SOW and REAP options.
TYPE: Integer
DEFAULT: 0
OPTIONS: n User-defined value.

- **METHOD_NAME**
  Which methods do you want to combine with the MFCC approach? It works with the REAP and SOW options. The same string should be used for the REAP and SOW options. This option accepts multiple method names since energies of several methods can be available in output files. Use the ";" character to specify multiple method names, such as [SCF;MP2;CCD].
  TYPE: String Array
  DEFAULT: [SCF]
  OPTIONS: User-defined available method names.

- **BSSE**
  Requests BSSE-corrected MFCC computations. This keyword is reserved for future versions; the current implementation does not yet support BSSE corrections in MFCC mode.
  TYPE: Boolean
  DEFAULT: FALSE
  OPTIONS: FALSE, TRUE.

21 AIMD Simulations: Ab-Initio Molecular Dynamics

Code Author: U. Bozkaya.

21.1 Overview

Molecular dynamics (MD) describes the time evolution of a set of nuclei on a potential energy surface. In classical MD that surface comes from a force field, and the quality of the simulation is bounded by the quality of the parameters. aimd removes that bound: the nuclei are propagated on the Born-Oppenheimer surface of whichever electronic-structure method the input asks for, so every step of the trajectory costs one analytic energy gradient at that level of theory.

The microcanonical (NVE), canonical (NVT) and isothermal-isobaric (NPT) ensembles are available; NPT is that of a cluster confined in a container, since a molecular code has no other way to give the volume a meaning. The propagator is velocity Verlet, with the thermostat applied for half a step on either side of the Verlet step so that the whole update stays symmetric. Six thermostats, rigid bond constraints, wavefunction extrapolation between steps and multilayer dynamics are supported, and a run can be stopped and continued from its restart file.

Everything is handled internally in atomic units. The input is given in the usual units — femtoseconds for the time step, picoseconds for the total duration, kelvin for the temperature — and converted once at start-up.

21.2 Statistical Ensembles

NVE. The number of atoms, the volume and the total energy are constant. This is adiabatic dynamics: kinetic and potential energy are exchanged and their sum is conserved. The conserved quantity reported in the output is $E = K + V$, and the drift of that number over the run is the honest measure of whether the time step was short enough.

NVT. The number of atoms, the volume and the temperature are constant. The energy is not conserved, by construction, so `aimd` instead reports

$$E_{\mathrm{cons}} = K + V + E_{\mathrm{bath}}, \tag{43}$$

where E_{bath} is the energy the thermostat has exchanged with the system since the start of the trajectory. This is the quantity to watch on an NVT run; without it there is no way to tell a well-integrated trajectory from a broken one.

The instantaneous temperature is

$$T = \frac{2K}{N_f k_B}, \qquad N_f = 3N - N_{\mathrm{c}} - N_{\mathrm{tr}}, \tag{44}$$

where N_{c} is the number of constrained bonds and N_{tr} the number of translational and rotational modes that are removed — six for a general molecule, five for a linear one, three for an atom. Both are reported in the output.

21.3 Thermostats

The choice is made with the `THERMOSTAT` option; the coupling time constant is `THERMOSTAT_TIMESCALE`, in femtoseconds.

- **CSVR** (default). Canonical sampling through velocity rescaling, Bussi, Donadio and Parrinello, *J. Chem. Phys.* **126**, 014101 (2007). Berendsen relaxation of the kinetic energy plus exactly the right amount of noise, so that the stationary distribution is the canonical one. It scales all velocities by one common factor, so it never disturbs bond constraints, and it has a well-defined conserved quantity. Recommended for production.
- **NOSE_HOOVER**. Nosé-Hoover chains, Martyna, Klein and Tuckerman, *J. Chem. Phys.* **97**, 2635 (1992), integrated with a Suzuki-Yoshida factorisation. Deterministic, time reversible and canonical. The chain length, the number of multiple time steps and the Suzuki-Yoshida order are set with `NOSE_HOOVER_LENGTH`, `NOSE_HOOVER_NC` and `NOSE_HOOVER_NYS`.

- **MASSIVE_NOSE_HOOVER**. One Nosé-Hoover chain per degree of freedom instead of one for the system. A single deterministic chain controls the total kinetic energy and nothing else, which for a molecule — few degrees of freedom, nearly harmonic modes that barely exchange energy — is not enough: started with the energy in one mode, a single chain can leave the modes at 130 and 485 K while reporting an average of 303 K. The massive variant equipartitions. Note that the trouble is with the *determinism* and not with being global: CSVR is also one common rescaling and equipartitions perfectly well, because its noise breaks the tori a deterministic chain gets stuck on. The price is that each Cartesian component gets its own factor, so this one does disturb bond constraints; with `CONSTRAINTS` set, prefer LANGEVIN.

- **LANGEVIN**. The friction and the noise are applied exactly as an Ornstein-Uhlenbeck half step on either side of the Verlet step (Bussi and Parrinello, *Phys. Rev. E* **75**, 056707 (2007)). Canonical, and it damps the internal dynamics as well, which makes it useful for equilibrating a badly prepared structure — at the cost of distorted time correlation functions.

- **ANDERSEN**. Stochastic collisions: a group of atoms has its velocities redrawn from the Maxwell-Boltzmann distribution with probability $1 - e^{-\nu \Delta t}$, with ν given by `ANDERSEN_FREQUENCY`. Canonical, but it destroys the momentum autocorrelation of the atoms it touches. The group is a whole constraint cluster rather than a single atom: redrawing one atom of a rigid pair and then projecting also moves its partner, and the result is not the constrained Maxwell-Boltzmann distribution — on a test system with three rigid bonds that samples 285 K against a target of 300.

- **BERENDSEN**. Weak coupling. The approach to the target temperature is smooth, but the distribution is not canonical. For equilibration.

- **RESCALE**. The kinetic energy is reset to its target value at every step. The temperature is pinned but does not fluctuate at all, so this is not the canonical ensemble either. For equilibration, and for reproducing the behaviour of older codes.

21.4 Rigid Bonds

The time step of a Born-Oppenheimer trajectory is set by the fastest motion in the system, which in any molecule containing hydrogen is the X–H stretch, with a period of about 9 fs. Holding those bonds rigid lets the step be raised by a factor of two to four, and since one step is one gradient, that is a factor of two to four in the cost of the whole simulation.

Constraints are requested with the `CONSTRAINTS` option: `H_BONDS` freezes the bonds to hydrogen, `ALL_BONDS` freezes every covalent bond. Bonds are identified from the initial geometry as pairs closer than 1.2 times the sum of the two covalent radii, and are held at the length they have in that geometry. The positions are corrected with SHAKE (Ryckaert, Ciccotti and Berendsen, *J. Comput. Phys.* **23**, 327 (1977)) and the velocities with RATTLE (Andersen, *J. Comput. Phys.* **52**, 24 (1983)).

Each constrained bond removes one degree of freedom, which is accounted for in the temperature. In a multilayer run a bond between two layers is never constrained.

21.5 Wavefunction Extrapolation

Two consecutive geometries in an MD run differ very little, so the converged orbitals of the previous steps are an excellent starting point for the next SCF. `aimd` keeps a short history of the occupied orbitals and predicts the next set with the always stable predictor-corrector (ASPC) of Kolafa, *J. Comput. Chem.* **25**, 335 (2004),

$$\mathbf{C}^{\mathrm{pred}} = \sum_{i=1}^{m} B_i \, \mathbf{C}_i \mathbf{C}_i^{\dagger} \mathbf{S} \, \mathbf{C}_1, \qquad B_1 = \frac{4m-2}{m+1}, \qquad B_i = (-1)^{i+1} i \frac{\binom{2m}{m-i}}{\binom{2m-2}{m-1}}, \tag{45}$$

after which the prediction is symmetrically orthonormalised against the overlap matrix. The projector $\mathbf{C}_i \mathbf{C}_i^{\dagger} \mathbf{S}$ puts every snapshot in the same orbital gauge before they are combined, which is what makes the sum meaningful.

The history holds `ASPC_ORDER`+2 snapshots and stores only the occupied columns, so its cost is $m \, n_{\mathrm{ao}} n_{\mathrm{occ}}$ doubles per subsystem. Setting `WF_EXTRAPOLATION PREVIOUS` keeps only the last set of orbitals, and `NONE` switches the whole mechanism off. In a multilayer run each subsystem keeps its own history.

21.6 Multilayer AIMD

The atom list can be split into a few contiguous layers, each with its own charge, multiplicity and level of theory, so that a reactive centre is propagated on, say, a CCSD(T) surface while its surroundings are propagated on a DFT one. The gradients of the layers are assembled into a single total force, and the dynamics, the thermostat, the constraints and the removal of translation and rotation all act on the whole system.

The geometry is given in a `$set_layer` block, with layers separated by `-` and each preceded by its own charge and multiplicity; the levels of theory are listed in `METHOD_PER_LAYER`. The potential is

$$E = \sum_{k} E_k[\mathrm{high}_k] + \lambda \left(E[\mathrm{low},\, \mathrm{all}] - \sum_{k} E_k[\mathrm{low}] \right), \tag{46}$$

with $\lambda = 0$ for `LAYER_INTERACTION NONE` (the default), in which the layers are electronically independent and the run costs L gradients per step, and $\lambda = 1$ for `TWO_BODY`, which adds the interlayer interaction evaluated at one common low level (`INTERACTION_METHOD`) and costs $2L + 1$ gradients per step. The force is the derivative of exactly this expression, so energy and gradient always belong to the same potential.

A layer whose multiplicity is not one is automatically run with an unrestricted reference; this can be overridden with `REFERENCE_PER_LAYER`.

21.7 The Isothermal-Isobaric Ensemble (NPT)

An NPT run needs a volume, and a volume is only a dynamical variable if the energy depends on it. `MacroQC` has no periodic electronic structure, so a box drawn around an isolated molecule enters no energy and no force; a barostat coupled to it has nothing to push against and the volume random-walks away. What is well defined for a molecular code is a *confined cluster*: a droplet held in a container whose wall is part of the Hamiltonian. That is what `ENSEMBLE NPT` does here, and it should be read as droplet NPT rather than the periodic NPT of a condensed-phase force-field code.

The container. A spherical, half-harmonic wall: nothing inside the radius, a stiff quadratic penalty outside it,

$$U_{\mathrm{wall}} = \sum_I \frac{k}{2}\,(r_I - R)^2 \quad \text{for } r_I > R, \qquad V = \frac{4}{3}\pi R^3, \tag{47}$$

with r_I measured from the container centre. The centre is the centre of mass of the input geometry and stays there, because the net force is projected out at every step; the droplet's centre of mass is therefore pinned to the centre of its container, which does no work and costs the three degrees of freedom that are already counted. A spherical wall exerts no torque, so the angular momentum behaves as before. The wall can also be switched on in NVE or NVT (`WALL SPHERICAL`), in which case the pressure and the volume are reported but nothing scales them.

Pressure. From the virial of the internal (quantum-mechanical) forces,

$$P = \frac{1}{3V}\left(2K + \sum_I \mathbf{R}_I \cdot \mathbf{F}_I\right). \tag{48}$$

The wall is the external constraint that balances this and is deliberately left out. The default `MOLECULAR` estimator uses the translational kinetic energy of the molecular centres of mass and the net force on each molecule, which is the pressure conjugate to the volume moves the barostat makes and is free of the intramolecular virial. `ATOMIC` uses every atom; it is the right choice for a single molecule, for which the molecular estimator vanishes identically. Molecules are identified once at start-up from the connected components of the covalent bond graph.

Barostats. `MTK` (default) is the extended-Lagrangian barostat of Martyna, Tobias and Klein: the container becomes a degree of freedom with velocity v_ϵ, mass $W = (N_f+3)k_B T\tau_P^2$ and its own Nosé-Hoover chain, propagated inside the Verlet step rather than applied to its result. Its force is

$$G_\epsilon = \left(1 + \frac{3}{N_f}\right) 2K + \sum_I \mathbf{R}_I \cdot \mathbf{F}_I - 3VP_{\mathrm{ext}}, \tag{49}$$

and the conserved quantity is a genuine extended Hamiltonian,

$$H' = K + U + \frac{p_\epsilon^2}{2W} + P_{\text{ext}}V + E_{\text{bath}}^{\text{part}} + E_{\text{bath}}^{\text{baro}}, \tag{50}$$

which is the practical advantage over the other two: an NPT trajectory can be checked the way an NVE one is. It costs no extra gradient. The barostat scales molecular centres of mass, so a molecule is carried rigidly and no bond is strained; N_f is then the number of molecular translational degrees of freedom, $3N_{\text{mol}} - 3$ with the centre of mass pinned, and at least two molecules are needed.

`MC` proposes a volume change every `BAROSTAT_FREQUENCY` steps, scales each molecule as a rigid body about the container centre, and accepts with

$$w = \Delta U + P_0 \Delta V - N k_B T \ln \frac{V'}{V}, \tag{51}$$

N being the number of independently scaled bodies — the number of molecules *minus one*, since the centre of mass is pinned to the centre of the container and the configuration integral carries V^{N-1}. This samples the isobaric ensemble exactly, and in the ideal-gas limit agrees with MTK. The trial is a gradient evaluation, so an accepted move already carries the forces the next step needs; at the default interval that is about four per cent of the run.

The size of the proposed move is adapted towards an acceptance between a quarter and three quarters while the run is equilibrating, and then held fixed. It is capped against the container volume as it was when the barostat first ran, not against the current one: a cap that follows the current volume makes the width of the proposal depend on the state and biases the sampling. That bias is negligible for the thousands of molecules a force-field code handles and is not negligible for the four to twenty of an *ab initio* droplet — in the ideal-gas limit, where the exact answer $\langle V \rangle = (N+1)k_B T/P$ is known, it costs 19% of the volume at $N = 4$. `BERENDSEN` relaxes the volume towards the target pressure with time constant `BAROSTAT_TIMESCALE` using the virial pressure that is already available, so it costs nothing extra, but — like the Berendsen thermostat — it gets the average right and the fluctuations wrong and belongs in equilibration — which is what it is for: the Monte Carlo barostat reaches equilibrium by diffusion and is slow when the container starts badly wrong, while the Berendsen one drives there monotonically. Equilibrate with Berendsen, produce with Monte Carlo.

An NPT run needs a thermostat as well; everything in the NVT section applies unchanged.

21.8 Output Files

- `<input>_aimd.dat` — one row per step: step number, time in fs, potential, kinetic, total and conserved energy in hartree, and the temperature in kelvin.
- `<input>_traj.xyz` — the trajectory in Ångstrom, with the energies and the temperature on the comment line of each frame. Written every `TRAJ_FREQUENCY` steps.

- `<input>_vel.xyz`, `<input>_frc.xyz` — velocities and forces in atomic units, if `PRINT_VELOCITIES` or `PRINT_FORCES` is set.
- `<input>_aimd.rst` — the restart file: positions, velocities, the internal state of the thermostat, the state of the random number generator and the running averages.

Everything is flushed as it is produced, so a trajectory that is killed part way through leaves usable output.

21.9 Restarting

Setting `MODE RESTART` continues from `<input>_aimd.rst`. The run picks up at the step the file records and continues to `MD_STEPS` (or to the number of steps implied by `MD_DURATION`), appending to the existing data and trajectory files. The forces are recomputed at the restored geometry rather than stored, which costs one extra gradient but makes it impossible to continue from a force belonging to a different geometry. A restarted trajectory reproduces the uninterrupted one to better than 10^{-10} E_h, including for the stochastic thermostats.

21.10 AIMD Module Options

- **ENSEMBLE**
  Statistical ensemble. NPT is the isothermal-isobaric ensemble of a cluster in a container; see the section above for what that does and does not mean.
  TYPE: String
  DEFAULT: NVE
  OPTIONS: NVE, NVT, NPT

- **MODE**
  New run or continuation.
  TYPE: String
  DEFAULT: NEW
  OPTIONS: NEW, RESTART

- **MD_STEP_SIZE**
  Integration time step, in femtoseconds. Use 0.5 fs or less without bond constraints, up to 1–2 fs with `CONSTRAINTS H_BONDS`.
  TYPE: Double
  DEFAULT: 0.5

- **MD_DURATION**
  Total simulated time, in picoseconds. Ignored if `MD_STEPS` is set.
  TYPE: Double
  DEFAULT: 1.0

- **MD_STEPS**
  Number of steps. If greater than zero it overrides `MD_DURATION`.
  TYPE: Integer
  DEFAULT: 0

- **EQUILIBRATION_STEPS**
  Length of the equilibration window. Those steps are excluded from the running averages, they are the window over which `TEMPERATURE_START` ramps and `THERMOSTAT_TIMESCALE_EQUIL` applies, and they are the only steps during which the Monte Carlo barostat adapts its volume step. With no ramp and no separate coupling the dynamics is unchanged and only the averaging is affected.
  TYPE: Integer
  DEFAULT: 0

- **TEMPERATURE**
  Target temperature in kelvin for an NVT or NPT run, and the temperature the initial velocities are drawn at in any ensemble.
  TYPE: Double
  DEFAULT: 298.15

- **TEMPERATURE_START**
  If non-negative, the target is ramped linearly from this temperature to `TEMPERATURE` over the first `EQUILIBRATION_STEPS` steps and held there afterwards. This is how a structure is heated from rest — set it to 0 together with `INITIAL_VELOCITIES ZERO` — and, with a start above the target, how it is annealed. While the target is moving there is no stationary distribution and E_{cons} is not conserved, which is what heating means; the `T_set/K` column of the data file records where the target was at each step.
  TYPE: Double
  DEFAULT: -1 (no ramp)

- **THERMOSTAT_TIMESCALE_EQUIL**
  If positive, the thermostat coupling time constant used during the equilibration window, after which `THERMOSTAT_TIMESCALE` takes over. A short constant here brings the system to the target quickly; the longer production value then leaves the dynamics alone.
  TYPE: Double
  DEFAULT: -1 (the same as production)

- **THERMOSTAT**
  Thermostat model. Ignored for NVE.
  TYPE: String
  DEFAULT: CSVR
  OPTIONS: CSVR, BERENDSEN, NOSE_HOOVER, MASSIVE_NOSE_HOOVER, LANGEVIN, ANDERSEN, RESCALE, NONE

- **THERMOSTAT_TIMESCALE**
  Thermostat coupling time constant τ, in femtoseconds. A value of the order of a

hundred femtoseconds couples weakly enough not to disturb the dynamics; a much smaller one drives the system to the target temperature faster but distorts it.
TYPE: Double
DEFAULT: 100.0

- **ANDERSEN_FREQUENCY**
  Collision frequency of the Andersen thermostat, in collisions per femtosecond per atom.
  TYPE: Double
  DEFAULT: 0.01

- **NOSE_HOOVER_LENGTH**
  Length of the Nosé-Hoover chain, global or massive.
  TYPE: Integer
  DEFAULT: 4

- **NOSE_HOOVER_NC**
  Number of multiple time steps per half step in the chain integration.
  TYPE: Integer
  DEFAULT: 1

- **NOSE_HOOVER_NYS**
  Order of the Suzuki-Yoshida factorisation.
  TYPE: Integer
  DEFAULT: 3
  OPTIONS: 1, 3, 5

- **INITIAL_VELOCITIES**
  How the initial velocities are generated.
  TYPE: String
  DEFAULT: MAXWELL
  OPTIONS: MAXWELL, THERMAL, ZERO, READ

  - MAXWELL : each Cartesian component is drawn from a Gaussian of variance $k_B T/m_i$.
  - THERMAL : every atom is given the equipartition speed $\sqrt{3k_B T/m_i}$ in a random direction. Reproducible, with no kinetic energy fluctuation.
  - ZERO : the trajectory starts at rest.
  - READ : read from `<input>_velocities.inp`, one line of three numbers per atom, in atomic units.

  In every case except READ the velocities are projected free of overall translation and rotation, corrected for the bond constraints, and only then scaled to the requested temperature.

- **RANDOM_SEED**
  Seed of the random number generator. Zero seeds it from the system entropy; any

positive value makes the run reproducible.
TYPE: Integer
DEFAULT: 0

- **REMOVE_COM_MOTION**
  Remove the net momentum from the initial velocities and the net force from the gradient at every step.
  TYPE: Boolean
  DEFAULT: TRUE

- **REMOVE_ROTATION**
  Remove the angular momentum from the initial velocities and the net torque from the gradient at every step.
  TYPE: Boolean
  DEFAULT: TRUE

- **CONSTRAINTS**
  Which bonds are held rigid.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, H_BONDS, ALL_BONDS

- **CONSTRAINT_TOLERANCE**
  Relative residual at which the SHAKE and RATTLE sweeps stop.
  TYPE: Double
  DEFAULT: 1.0e-10

- **CONSTRAINT_MAX_ITER**
  Maximum number of SHAKE/RATTLE sweeps.
  TYPE: Integer
  DEFAULT: 500

- **WF_EXTRAPOLATION**
  How the SCF guess is carried from one step to the next.
  TYPE: String
  DEFAULT: ASPC
  OPTIONS: ASPC, PREVIOUS, NONE

- **ASPC_ORDER**
  Order of the ASPC extrapolation. The history holds `ASPC_ORDER`+2 snapshots.
  TYPE: Integer
  DEFAULT: 3

- **TRAJ_FREQUENCY**
  Write a trajectory frame every this many steps.
  TYPE: Integer
  DEFAULT: 1

- **RESTART_FREQUENCY**
  Rewrite the restart file every this many steps.
  TYPE: Integer
  DEFAULT: 1

- **PRINT_VELOCITIES**, **PRINT_FORCES**
  Also write the velocities and the forces as xyz files.
  TYPE: Boolean
  DEFAULT: FALSE

- **WALL**
  Confining boundary. Required for NPT, where it is switched on automatically if left at NONE; optional in NVE and NVT, where it makes the pressure and the volume well defined without acting on them.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, SPHERICAL

- **WALL_RADIUS**
  Container radius in Ångstrom. Zero means "large enough to hold the input geometry": the radius is then the largest distance from the centre of mass to an atom, plus `WALL_MARGIN`.
  TYPE: Double
  DEFAULT: 0.0

- **WALL_MARGIN**
  Margin in Ångstrom used when the radius is chosen automatically.
  TYPE: Double
  DEFAULT: 2.0

- **WALL_FORCE_CONSTANT**
  Stiffness of the half-harmonic wall, in $E_\mathrm{h}/\mathrm{bohr}^2$. The default lets an atom penetrate about 0.05 Å at 300 K.
  TYPE: Double
  DEFAULT: 0.1

- **PRESSURE**
  Target pressure in bar.
  TYPE: Double
  DEFAULT: 1.0

- **BAROSTAT**
  Barostat model.
  TYPE: String
  DEFAULT: MTK
  OPTIONS: MTK, MC, BERENDSEN

  - MTK : extended-Lagrangian barostat with a conserved quantity, costing no extra gradient. Needs at least two molecules.
  - MC : Monte Carlo, exact in distribution, one extra gradient every `BAROSTAT_FREQUENCY` steps.
  - BERENDSEN : weak coupling, free, for equilibration only.

- **EQUILIBRATION_STEPS and the MC barostat**
  The Monte Carlo volume step adapts towards a sensible acceptance only while the run is not being sampled, because a proposal whose width depends on the state biases the sampling. With `EQUILIBRATION_STEPS` left at zero it therefore never adapts and stays at one per cent of the initial container: correct, but possibly very slow. The run says so in the output.

- **BAROSTAT_FREQUENCY**
  Number of MD steps between barostat moves, for MC and BERENDSEN; each move costs one extra gradient. Ignored by MTK, which is continuous.
  TYPE: Integer
  DEFAULT: 25

- **BAROSTAT_TIMESCALE**
  Barostat coupling time constant in femtoseconds: the relaxation time for BERENDSEN, and the frequency that sets the barostat mass and its chain for MTK.
  TYPE: Double
  DEFAULT: 1000.0

- **COMPRESSIBILITY**
  Isothermal compressibility used by the Berendsen barostat, in bar^{-1}. The default is that of liquid water.
  TYPE: Double
  DEFAULT: 4.5e-5

- **PRESSURE_ESTIMATOR**
  What a volume move scales, and therefore which virial the pressure is built from — the two must be conjugate, so one option sets both. MOLECULAR scales molecular centres of mass rigidly and uses the molecular virial; it is right for a fluid of molecules and is the only choice compatible with bond constraints. ATOMIC scales the atoms individually and uses the atomic virial; it is what makes NPT meaningful for a *single* molecule under hydrostatic pressure, for which the molecular virial of a pinned centre of mass vanishes identically. ATOMIC and `CONSTRAINTS` together are refused, because scaling atoms stretches the constrained bonds.
  TYPE: String
  DEFAULT: MOLECULAR
  OPTIONS: MOLECULAR, ATOMIC

- **MULTILAYER**
  Activate multilayer dynamics. The geometry must then be given in a `$set_layer`

block.
TYPE: Boolean
DEFAULT: FALSE

- **METHOD_PER_LAYER**
  Electronic-structure method for each layer. One entry per layer, or a single entry used for all of them.
  TYPE: String Array
  DEFAULT: [SCF]

- **REFERENCE_PER_LAYER**
  Reference wavefunction for each layer. The default AUTO uses the global `REFERENCE`, switching to UHF for any layer whose multiplicity is not one.
  TYPE: String Array
  DEFAULT: [AUTO]

- **LAYER_INTERACTION**
  Interlayer correction.
  TYPE: String
  DEFAULT: NONE
  OPTIONS: NONE, TWO_BODY

- **INTERACTION_METHOD**
  Level of theory at which the interlayer correction is evaluated.
  TYPE: String Array
  DEFAULT: [SCF]

- **PRINT_LEVEL**
  0 writes only the data files, 1 also writes a per-step summary to the output, 2 and above add the full electronic-structure output of every step.
  TYPE: Integer
  DEFAULT: 1

21.11 Input Examples

A microcanonical trajectory:

```
$set_molecule H2O
    0 1
    O   0.00000   0.00000  -0.06577
    H   0.00000  -0.75906   0.52195
    H   0.00000   0.75906   0.52195
$end_molecule

$set_globals
    basis cc-pvdz
```

```
    aux_basis_scf cc-pvdz-jkfit
    method scf
    reference rhf
    jobtype aimd
    memory 2000
$end_globals

$set_aimd
    ensemble nve
    md_step_size 0.5
    md_duration 1.0
    temperature 300.0
$end_aimd
```

A canonical trajectory with rigid X–H bonds and a longer time step:

```
$set_aimd
    ensemble nvt
    thermostat csvr
    thermostat_timescale 100.0
    constraints h_bonds
    md_step_size 1.0
    md_duration 10.0
    temperature 300.0
    equilibration_steps 500
    wf_extrapolation aspc
    aspc_order 3
    traj_frequency 5
$end_aimd
```

Multilayer dynamics, with the first layer at MP2 and the second at SCF, and the interlayer interaction added at the SCF level:

```
$set_layer
0 1
O    -1.551007   -0.114520    0.000000
H    -1.934259    0.762503    0.000000
H    -0.599677    0.040712    0.000000
--
0 1
O     1.350625    0.111469    0.000000
H     1.680398   -0.373741   -0.758561
H     1.680398   -0.373741    0.758561
$end_layer

$set_globals
    basis cc-pvdz
```

```
    aux_basis_scf cc-pvdz-jkfit
    aux_basis_corr cc-pvdz-ri
    method scf
    reference rhf
    jobtype aimd
    memory 2000
$end_globals

$set_aimd
    ensemble nve
    multilayer true
    method_per_layer [MP2;SCF]
    layer_interaction two_body
    interaction_method [SCF]
    md_step_size 0.5
    md_duration 0.1
    temperature 300.0
$end_aimd
```

Continuing a run that was stopped:

```
$set_aimd
    ensemble nvt
    mode restart
    thermostat csvr
    thermostat_timescale 100.0
    md_step_size 1.0
    md_duration 20.0
    temperature 300.0
$end_aimd
```

A droplet in the isothermal-isobaric ensemble:

```
$set_aimd
    ensemble npt
    thermostat csvr
    thermostat_timescale 100.0
    barostat mtk
    barostat_timescale 1000.0
    pressure 1.0
    wall spherical
    wall_margin 2.0
    md_step_size 0.5
    md_duration 20.0
    temperature 300.0
    equilibration_steps 1000
$end_aimd
```

22 Solvation

Code Author: U. Bozkaya.

22.1 Overview

A continuum solvation model replaces the solvent by a structureless dielectric, which is far cheaper than sampling explicit solvent molecules and is often accurate enough [78]. `MacroQC` implements several such models. All of them share the same cavity machinery and the same coupling to the SCF, and all of them split the solvation free energy into an electrostatic part, obtained self-consistently, and a non-electrostatic part:

$$\Delta G_{solv} \;=\; \Delta G_{elec} \;+\; G_{non-elec}. \tag{52}$$

What distinguishes them is how they treat each half.

Solvation is switched on with `SOLVATION TRUE` in the `SCF` block, and the model is chosen with `SOLVATION_MODEL`:

`SOLVATION_MODEL`	electrostatics	non-electrostatics	options block
`GB`	generalized Born, atomic charges	SMD surface tensions	`GB`
`SM12`	generalized Born, CM5 charges	SM12 surface tensions	`GB`
`COSMO`	conductor-like screening	—	`COSMO`
`CPCM`	conductor-like, surface charges	optional cavitation	`PCM`
`IEFPCM`	integral equation formalism	optional cavitation	`PCM`
`SMD`	COSMO on SMD radii	atomic surface tensions	`SMD`
`CMIRS`	SS(V)PE on an isodensity cavity	four density-based terms	`CMIRS`
`COSMORS`	from a COSMO surface, statistical	—	`COSMORS`
`VASPSOL`	generalized Poisson, no cavity	quantum surface area	`VASPSOL`

For example,

```
$set_scf
  solvation true
  solvation_model iefpcm
$end_scf

$set_pcm
  epsilon 78.3553
$end_pcm
```

`GB`, `SM12`, `COSMO`, `CPCM`, `IEFPCM` and `VASPSOL` run inside a single SCF and may be combined with any jobtype. `SMD`, `CMIRS` and `COSMORS` need more than one SCF run or a post-processing step, so they take over the job; they may also be requested directly as `JOBTYPE SMD`, `CMIRS`

or COSMORS. COSMORS is available for JOBTYPE ENERGY alone. SMD and CMIRS take derivative jobtypes as well, SMD analytically and CMIRS by finite difference.

Every model works with the RHF, UHF, ROHF, RKS and UKS references. Analytic gradients are available for COSMO, for C-PCM and IEF-PCM, for the cavitation term, for the SMD and SM12 non-electrostatic terms, and for the generalized-Born model and SM12 as a whole, for SMD and for VASPsol. They are not available for COSMO-RS, and not for CMIRS, whose gradient is numerical. The generalized-Born gradient needs a charge model that can be differentiated, which rules out the two intrinsic-orbital variants; CM5, Mulliken and Löwdin all work, with any of the RHF, UHF, ROHF, RKS and UKS references.

22.2 The Cavity

Every model but the generalized-Born one and VASPsol lives on a discretized cavity surface. MacroQC builds it in the GEPOL spirit [79]: one sphere per atom, each tessellated with a pentakis-dodecahedron or Lebedev grid, with the tesserae that fall inside a neighbouring sphere removed.

With a hard cutoff a tessera is either inside a neighbour or it is not, so the exposed area is a step function of the geometry and has no derivative. The switching function of York and Karplus [80, 81] removes the steps, and the modules that need a gradient turn it on themselves.

22.2.1 GEPOL Module Options

- **SURFACE_TYPE**
  TYPE: STRING
  DEFAULT: SAS
  OPTIONS: SAS, VDW, SES
  DESCRIPTION: which surface to build. VDW is the bare van der Waals surface of the atomic spheres and is what PCM uses; SAS grows every sphere by the probe radius; SES additionally inserts spheres in the crevices a probe cannot reach.

- **RADIUS_TYPE**
  TYPE: STRING
  DEFAULT: BONDI (GEPOL block); UFF (PCM block)
  OPTIONS: BONDI, BONDI_PCM, ALVAREZ, SMD, UFF, COV_PLUS_CONST
  DESCRIPTION: which table of atomic radii to build the spheres from. BONDI is Bondi's set [82] with the Mantina extensions [83]; BONDI_PCM is the same with Rowland and Taylor's hydrogen radius, which is what the PCM literature usually means. UFF is the van der Waals set of the universal force field [84], $x_i/2$, which puts oxygen at 1.750 and hydrogen at 1.443 Å – not the generic van der Waals radii, which are 1.52 and 1.20. The PCM cavity defaults to UFF scaled by 1.1, which is what the most-used implementations of the model use.

- **TESS_TYPE**
  TYPE: STRING
  DEFAULT: GEPOL
  OPTIONS: GEPOL, LEBEDEV
  DESCRIPTION: how each sphere is tessellated. GEPOL subdivides a pentakis dodecahedron, which is what the original code does; LEBEDEV uses a spherical quadrature grid instead.

- **NDIV**
  TYPE: INTEGER
  DEFAULT: 4
  OPTIONS: 1–5
  DESCRIPTION: the subdivision level of the pentakis dodecahedron. A sphere carries $60n^2$ tesserae, so the cost of a PCM calculation rises steeply with it; `libpcm` lowers the default to 3 for its own cavity.

- **LEBEDEV_L**, **LEBEDEV_L_H**
  TYPE: INTEGER
  DEFAULT: 29
  DESCRIPTION: the Lebedev order for heavy atoms and for hydrogen, when TESS_TYPE is LEBEDEV.

- **SWITCH_TYPE**
  TYPE: STRING
  DEFAULT: NONE
  OPTIONS: NONE, SWIG
  DESCRIPTION: NONE removes a buried tessera outright; SWIG fades it out smoothly so that the area is differentiable. A gradient needs SWIG and turns it on.

- **SCALE_VDW**
  TYPE: DOUBLE
  DEFAULT: 1.0
  DESCRIPTION: a uniform factor applied to every radius.

- **SOLVENT_RADIUS**
  TYPE: DOUBLE
  DEFAULT: 1.4
  DESCRIPTION: the probe radius in Å, used by SAS and by the spheres GEPOL adds for SES.

22.3 COSMO

The conductor-like screening model [85] replaces the solvent by a conductor, which answers the solute by placing a charge on the cavity boundary that cancels the solute potential there exactly, and recovers the response of a real dielectric by scaling that answer by $f(\varepsilon) =$

$(\varepsilon - 1)/(\varepsilon + \frac{1}{2})$. The approximation is good because f is already close to one for any polar solvent, and it avoids the ill-conditioning that afflicts the general surface-charge equations.

22.3.1 COSMO Module Options

- **EPSILON**
  TYPE: DOUBLE
  DEFAULT: -1
  DESCRIPTION: the relative permittivity of the solvent. A negative value means the conductor limit, $f = 1$.

- **NPPA**
  TYPE: INTEGER
  DEFAULT: 1082
  DESCRIPTION: points per atom on the fine basis grid, which decides which parts of the surface are exposed. Must be of the form $10 \cdot 3^k 4^l + 2$.

- **NSPA**
  TYPE: INTEGER
  DEFAULT: 92
  DESCRIPTION: the number of surface segments per heavy atom; the number for hydrogen follows from it.

- **CAVITY**
  TYPE: INTEGER
  DEFAULT: 1
  DESCRIPTION: 1 closes the seams where two spheres intersect with extra segments, 0 leaves them open.

- **RSOLV**
  TYPE: DOUBLE
  DEFAULT: -1
  DESCRIPTION: the probe radius in Å used to decide what is exposed. A negative value takes the optimized value of 1.30.

- **ROUTF**
  TYPE: DOUBLE
  DEFAULT: 0.85
  DESCRIPTION: the outlying-charge correction uses a second surface at $R + \mathrm{ROUTF} \cdot \mathrm{RSOLV}$.

- **OC_CORRECTION**
  TYPE: BOOL
  DEFAULT: TRUE
  DESCRIPTION: recover the part of the electron density that has leaked outside the cavity, which the surface charges would otherwise miss.

- **RADIUS_TYPE**
  TYPE: STRING
  DEFAULT: COSMO
  OPTIONS: COSMO, SMD
  DESCRIPTION: Klamt's optimized radii, or the intrinsic Coulomb radii SMD is defined on. SMD sets this itself.

- **RADII_ATOMS**, **RADII_VALUES**
  TYPE: ARRAY
  DESCRIPTION: per-element overrides in Å, for instance `radii_atoms [O, H]` together with `radii_values [1.42; 1.30]`.

- **WRITE_COSMO_FILE**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: write the `.cosmo` file that a later COSMO-RS calculation reads.

- **BOHR_CONVENTION**
  TYPE: STRING
  DEFAULT: COSMO
  OPTIONS: COSMO, CODATA
  DESCRIPTION: the radii were fitted with 1 bohr = 0.529177249 Å and every COSMO implementation still uses it; CODATA switches to the modern value and changes the energy in the last digits.

- **CHECK**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: run self-tests of the electrostatic-potential and field integrals the model rests on.

22.4 PCM: C-PCM and IEF-PCM

The polarizable continuum model [86, 78] represents the solvent response by an apparent charge density on the cavity boundary. Two boundary-integral operators are discretized by collocation on the tesserae, and the members of the family differ only in the linear system that turns the solute potential $\boldsymbol{\varphi}$ into the charges $\boldsymbol{q}$:

$$\text{C-PCM} \;:\; \boldsymbol{q} \;=\; -f(\varepsilon)\,\boldsymbol{S}^{-1}\boldsymbol{\varphi}, \tag{53}$$

$$\text{IEF-PCM} \;:\; \boldsymbol{q} \;=\; -\boldsymbol{T}^{-1}\boldsymbol{R}\,\boldsymbol{\varphi}. \tag{54}$$

C-PCM [87] keeps only the single-layer operator and is the cheaper and better-conditioned of the two; GCOSMO [88] is the same equation with a different scaling factor. IEF-PCM [89] keeps both operators and, for an isotropic solvent, is identical to Chipman's SS(V)PE [90].

For a small cavity the response matrix is formed explicitly and every SCF iteration is then a single matrix-vector product. Above a few thousand tesserae that matrix no longer fits in memory, and the solver switches to a matrix-free iterative scheme that stores nothing at all.

22.4.1 PCM Module Options

- **GAUSSIAN_KERNEL**
  TYPE: BOOL
  DEFAULT: TRUE
  DESCRIPTION: put a normalized spherical Gaussian on each tessera instead of a point charge, so that the single-layer operator becomes $\mathrm{erf}(\zeta_{ij} r_{ij})/r_{ij}$, with $\zeta_{ij} = \zeta_i \zeta_j / \sqrt{\zeta_i^2 + \zeta_j^2}$ and $\zeta_i = \zeta(n)/\sqrt{a_i}$ for a_i the bare tessera area and $\zeta(n)$ the York–Karplus exponent of the Lebedev order. The self-interaction becomes $(2/\sqrt{2\pi})\zeta_i$ divided by the switching factor. Together with `SWITCH_TYPE SWIG` this is the kernel Q-Chem uses, and with it the two codes agree on ΔG to 0.026 kcal/mol on average over a nine-molecule set, against 0.069 with point charges. Only the single-layer operator is blurred; D keeps the sharp kernel. On by default, together with the Lebedev tessellation, `SWITCH_TYPE SWIG` and `SUM_RULE`, because that is the cavity the most-used implementations of the model default to; with all of them the two agree on ΔG to 0.03 kcal/mol out of the box, each program using nothing but its own defaults. Switched off automatically when the cavity is supplied from outside, as it is for CMIRS, since a supplied surface carries no bare areas or switching factors to build the Gaussian widths from.

- **METHOD**
  TYPE: STRING
  DEFAULT: IEFPCM
  OPTIONS: IEFPCM, CPCM, GCOSMO
  DESCRIPTION: which member of the family to use.

- **EPSILON**
  TYPE: DOUBLE
  DEFAULT: 78.3553
  DESCRIPTION: the relative permittivity of the solvent; the default is water.

- **CPCM_CORRECTION**
  TYPE: DOUBLE
  DEFAULT: 0.0
  DESCRIPTION: the x in $f(\varepsilon) = (\varepsilon - 1)/(\varepsilon + x)$. Zero is the usual C-PCM choice; $\frac{1}{2}$ reproduces Klamt's COSMO scaling. GCOSMO forces it to zero. It has no effect on IEF-PCM.

- **DIAGONAL_FACTOR**
  TYPE: DOUBLE
  DEFAULT: 1.0694

DESCRIPTION: the empirical constant k in $S_{ii} = k\sqrt{4\pi/a_i}$, which sets the self-interaction of a tessera. Do not change it without reason: the parameterizations were fitted with it.

- **HERMITIZE**
  TYPE: BOOL
  DEFAULT: TRUE
  DESCRIPTION: symmetrize the IEF response matrix. The energy is unchanged, but the individual charges are not, and with a symmetric matrix the Fock contribution is the exact derivative of the energy so that the SCF is variational.

- **SUM_RULE**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: take the diagonal of the double-layer operator from the exact relation $\sum_j D_{ij}a_j = -2\pi$ rather than from the collocation formula. It is forced when the cavity has no spheres behind it, as for CMIRS, and it rules out the matrix-free solver.

- **RADIUS_TYPE**
  TYPE: STRING
  DEFAULT: BONDI_PCM
  OPTIONS: BONDI_PCM, BONDI, ALVAREZ, SMD, UFF
  DESCRIPTION: the table the cavity spheres come from.

- **SCALE_RADII**
  TYPE: DOUBLE
  DEFAULT: 1.0
  DESCRIPTION: a uniform factor on the radii. Values around 1.2 are common in the older PCM literature.

- **RADII_ATOMS**, **RADII_VALUES**
  TYPE: ARRAY
  DESCRIPTION: per-element overrides in Å. The hydrogen radius is a convention rather than a measurement: BONDI_PCM uses 1.10 after Rowland and Taylor, whereas Bondi's own value is 1.20, and on water the two differ by 1.4 kcal/mol. Fix it explicitly before comparing with another program.

- **SOLVER**
  TYPE: STRING
  DEFAULT: AUTO
  OPTIONS: AUTO, DIRECT, ITERATIVE
  DESCRIPTION: DIRECT forms and inverts the response matrix, which is much the faster where it fits; ITERATIVE forms nothing and solves by conjugate gradients or BiCGSTAB, which is the only route for a large cavity. AUTO leaves DIRECT when the matrices would take more than half of MEMORY.

- **SOLVER_CONVERGENCE**
  TYPE: DOUBLE
  DEFAULT: 10^{-10}
  DESCRIPTION: the relative residual the iterative solver stops at.

- **SOLVER_MAXITER**
  TYPE: INTEGER
  DEFAULT: 500
  DESCRIPTION: an upper bound on the iterations of a single solve.

- **GAS_PHASE**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: run the solute in vacuum first, so that the solvation free energy can be reported against it. Without it what is printed is $G_{pol} + G_{CDS}$ at the solvated density, which is missing the work of distorting the solute, $E[\rho_{sol}] - E[\rho_{gas}]$ — always positive, and a third of a kcal/mol on acetone in water. With it `MacroQC` prints the full breakdown,

$$\Delta G_{ENP} \;=\; \Big(E[\rho_{sol}] - E[\rho_{gas}]\Big) \;+\; \Delta G_{pol}, \qquad \Delta G_{solv} \;=\; \Delta G_{ENP} \;+\; G_{CDS}. \tag{55}$$

  Energy jobs only; a derivative job wants the free energy of the solvated molecule, not a difference of two states.

- **GRADIENT**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: compute the analytic derivative. It is implied by any jobtype that takes one, and it turns on the switching function of the cavity, without which the exposed area has no derivative.

- **SMOOTH_SURFACE**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: use switched areas in an energy-only run, which is what a finite difference of the energy has to be compared against.

- **CAVITATION**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: add the work of opening the cavity, from scaled-particle theory [91]. It is always positive and can be several kcal/mol.

- **SOLVENT_RADIUS**
  TYPE: DOUBLE
  DEFAULT: 1.385
  DESCRIPTION: the hard-sphere radius of the solvent in Å, used only by the cavitation term.

- **TEMPERATURE**
  TYPE: DOUBLE
  DEFAULT: 298.15
  DESCRIPTION: in K, for the cavitation term.

- **PRESSURE**
  TYPE: DOUBLE
  DEFAULT: 1.0
  DESCRIPTION: in bar, for the pressure-volume part of the cavitation work.

- **SOLVENT_RHO**
  TYPE: DOUBLE
  DEFAULT: 0.0
  DESCRIPTION: the solvent number density in Å^{-3}. A non-positive value takes an empirical correlation for liquid water at the given temperature.

22.5 SMD

The SMD universal solvation model [92] splits the solvation free energy into a bulk-electrostatic part, taken from two SCF runs with the continuum switched off and on, and a short-range part built from empirical atomic surface tensions on a solvent-accessible surface. The two halves were fitted together and neither is meaningful alone. A solvent enters through six descriptors, so SMD covers a very large number of solvents with one parameter set.

22.5.1 SMD Module Options

- **SOLVENT**
  TYPE: STRING
  DEFAULT: WATER
  DESCRIPTION: the solvent name; the list is at the end of this section.

- **SOLVENT_PARAMS**
  TYPE: STRING
  DEFAULT: MN21
  OPTIONS: MN21, SM8
  DESCRIPTION: which set of solvent descriptors to use. MN21 is the later revision.

- **CDS_TYPE**
  TYPE: STRING
  DEFAULT: SMD
  OPTIONS: SMD, SM8AD
  DESCRIPTION: which parameterization of the molecular surface tension to use.

- **SOLVENT_RADIUS**
  TYPE: DOUBLE

DEFAULT: 0.4
DESCRIPTION: the probe in Å the surface tensions were fitted with. It is not a physical solvent radius and changing it invalidates the parameterization.

- **GRADIENT**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: also compute the analytic derivative of the solvation free energy, which is the difference of the two SCF gradients plus the derivative of the surface-tension term.

- **SMOOTH_SURFACE**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: switched areas without a gradient. It moves the surface-tension term by a few thousandths of a kcal/mol from the fitted convention.

22.6 CMIRS

The composite method for the implicit representation of solvent [93, 94, 95] adds four terms to the electrostatics, all of them read off the solute electron density rather than off atomic surface tensions, so that no atom typing is needed anywhere:

$$G_{non-elec} \;=\; A\,E_{disp} \;+\; B\,E_{exch} \;+\; C\,|F_{min}|^{\gamma} \;+\; D\,F_{max}^{\gamma}. \tag{56}$$

E_{exch} is the integral of $|\nabla\rho|$ over the region outside the cavity and stands for Pauli repulsion; E_{disp} is a damped London expression built on the local polarizability of VV09 [96]; and F_{min} and F_{max} are the extreme values of the outward normal electric field on the cavity, standing for the strongest hydrogen bond the solute can donate and accept. The electrostatics are SS(V)PE on the same isodensity surface.

The model is parameterized for five solvents at two isodensity contours. The contour is part of the parameterization and not a convergence parameter: nothing between the two tabulated values is defined.

The cavity is built from one centre, by shooting the rays of a Lebedev grid and taking the outermost crossing of the contour along each. That assumes the surface is star shaped about the centre, and the assumption is about the solute, not about the settings. It holds for compact solutes: water, n-butane and a hydrogen-bonded water dimer all use every ray, and the two ways of evaluating the non-electrostatic terms agree on them to a few hundredths of a kcal/mol. It fails for a solute that separates into pieces. Two water molecules seven Ångstroms apart leave 82 of 590 rays with no crossing at all – they pass through the gap – and both the cavity and everything computed on it, the electrostatics included, are then meaningless. The tessera count is printed against the ray count for exactly this reason and is worth reading.

Between the two there is a middle ground to be careful in. At four Ångstroms every ray still crosses, and yet F_{min} comes out -0.048 on the rays against -0.062 on the cubic grid,

which is fifteen kcal/mol in the donor term. The rays describe the outer envelope and miss the concave neck between the molecules, which is where the normal field is strongest, and refining the ray count does not help: F_{min} and F_{max} are extrema over a sample of a surface, so they only grow more extreme as the sample grows. For a compact solute they are stable to four decimals. For a concave one they are not a converged quantity in either evaluation, and a number from CMIRS should be read with that in mind.

22.6.1 CMIRS Module Options

- **SOLVENT**
  TYPE: STRING
  DEFAULT: WATER
  OPTIONS: WATER, BENZENE, CYCLOHEXANE, DMSO, ACETONITRILE, NONE
  DESCRIPTION: which tabulated parameter set to use. NONE means the coefficients below must all be supplied by hand.

- **ION_REFERENCE**
  TYPE: STRING
  DEFAULT: KELLY
  OPTIONS: KELLY, FAWCETT
  DESCRIPTION: DMSO and acetonitrile were fitted twice, against two experimental values of the proton solvation free energy. Only these two solvents are affected.

- **ISO_DENSITY**
  TYPE: DOUBLE
  DEFAULT: 0.001
  OPTIONS: 0.0005 or 0.0010
  DESCRIPTION: the contour of the electron density that defines the cavity, in a.u. Each value is a separate parameterization.

- **A**, **B**, **C**, **D**, **GAMMA**
  TYPE: DOUBLE
  DESCRIPTION: the coefficients above, which override the tabulated values. A is negative because dispersion is attractive; C and D are zero for solvents that do not hydrogen bond.

- **SOLVENT_RHO**
  TYPE: DOUBLE
  DESCRIPTION: the solvent's average *electron* density in bohr^{-3}, not its number density. Water is ten electrons at 0.0334 molecules per Å^3, which is 0.05.

- **DELTA**
  TYPE: DOUBLE
  DEFAULT: 7.0
  DESCRIPTION: the damping length of the dispersion kernel in bohr, which keeps the

$1/r^6$ integrand finite. It was fixed rather than optimized in the published parameterization.

- **ISODENSITY_CAVITY**
  TYPE: BOOL
  DEFAULT: TRUE
  DESCRIPTION: put the electrostatics on the isodensity surface, as the model intends, rather than on the van der Waals cavity. Turning it off is worth several kcal/mol and is only useful for testing.

- **CAVITY_LEBEDEV_L**
  TYPE: INTEGER
  DEFAULT: 41
  DESCRIPTION: the order of the single-center Lebedev grid whose rays locate the isodensity surface.

- **SPACING**, **PADDING**
  TYPE: DOUBLE
  DEFAULT: 0.25 and 4.0
  DESCRIPTION: the cubic grid the density is sampled on, in Å, and how far it extends beyond the outermost nucleus.

- **NRADIAL**, **LEBEDEV_L**
  TYPE: INTEGER
  DEFAULT: 75 and 29
  DESCRIPTION: the atom-centered quadrature the dispersion integral is taken on.

- **NONELS_GRID**
  TYPE: STRING
  DEFAULT: CUBE
  OPTIONS: CUBE, RAY
  DESCRIPTION: which of the model's two ways of evaluating the four non-electrostatic terms to use. CUBE is the cubic grid above, and is what the published numbers were produced with; it decides which region a point belongs to by a test on the density there, so the energy steps as a point crosses the contour. RAY writes the same terms on the rays of the isodensity surface, where the boundary is resolved exactly: the exchange integral by Gauss–Laguerre along each ray, the dispersion integral in closed form as a sum over the solid-angle elements. The two agree to a few hundredths of a kcal/mol on a compact solute and RAY is the faster. A derivative jobtype selects RAY by itself, because CUBE cannot be differenced well.

- **GAULAG_N**
  TYPE: INTEGER
  DEFAULT: 20
  DESCRIPTION: Gauss–Laguerre points along each ray, for `NONELS_GRID RAY`. The rule is exact for a density falling exponentially, which is what the substitution takes out, so twenty is generous.

- **FIELD_ALL_CROSSINGS**
  TYPE: BOOLEAN
  DEFAULT: FALSE
  DESCRIPTION: whether the field extrema see every crossing of the contour along a ray or only the outermost one, which is the one that bounds the cavity. They differ only where a ray crosses more than once, which is where the star-shaped cavity is already doubtful; the default is one point per direction, as in the model's own implementation.

22.7 COSMO-RS

COSMO-RS [97, 98] turns a COSMO surface into a chemical potential by treating the screening-charge densities of the species in a mixture statistically. It needs no dielectric constant: the solvent is described by its own COSMO surface rather than by a permittivity, which is why the method can treat mixtures. It reads `.cosmo` files written by earlier COSMO runs.

22.7.1 COSMO-RS Module Options

- **COSMO_FILES**
  TYPE: LINE
  DESCRIPTION: the `.cosmo` files of every species in the mixture, separated by spaces and quoted so that the paths keep their case.

- **MOLE_FRACTIONS**
  TYPE: ARRAY
  DESCRIPTION: the composition, in the same order as the files. Omitting it puts the first species at infinite dilution in the rest.

- **PARAMETERIZATION**
  TYPE: STRING
  DEFAULT: TURBOMOLE
  OPTIONS: TURBOMOLE, ORCA
  DESCRIPTION: which published set of interaction parameters to use; they go with the level of theory the `.cosmo` files were produced at.

- **TEMPERATURE**
  TYPE: DOUBLE
  DEFAULT: 298.15
  DESCRIPTION: in K.

- **REFERENCE_STATE**
  TYPE: STRING
  DEFAULT: COSMO
  OPTIONS: COSMO, PURE

DESCRIPTION: whether the chemical potential is reported relative to the ideal conductor or to the pure liquid of the species itself.

- **MAXITER**, **CONVERGENCE**
  TYPE: INTEGER, DOUBLE
  DEFAULT: 1000 and 10^{-6}
  DESCRIPTION: the self-consistent solution of the COSMOSPACE equations.

22.8 Class-IV Charges

Solvation models driven by partial charges are only as good as the charges, and Mulliken or Löwdin populations are too basis-dependent to carry a parameterization. `MacroQC` provides Hirshfeld charges [99], whose promolecule of spherically averaged free atoms comes from the SAD guess, and the CM5 model [100] built on top of them. CM5 corrects the Hirshfeld charges with a pairwise, bond-order-weighted map whose parameters were fitted once against experimental dipole moments and carry no index for the basis set or the functional.

22.8.1 Options

- **SCF CM5_CHARGES**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: compute and print both the Hirshfeld and the CM5 charges after the SCF, together with the dipole moment each set reproduces.

- **SMX NRADIAL**, **SMX LEBEDEV_L**
  TYPE: INTEGER
  DEFAULT: 99 and 41
  DESCRIPTION: the quadrature the Hirshfeld partition is integrated on. The charges sum to the total charge exactly whatever the grid, so this controls accuracy and not consistency.

22.9 The Generalized-Born Model

`MacroQC` also provides a generalized-Born model [101], in which the electrostatics are built from atomic partial charges and effective Born radii rather than from surface charges, and the non-electrostatic term uses the SMD surface tensions. It is the cheapest of the models here: there is no cavity, no surface charge and no linear system to solve.

The free energy the model defines is

$$G \;=\; E_{SCF}[D] \;+\; \Delta G_{pol}[q(D)] \;+\; G_{CDS}, \tag{57}$$

$$\Delta G_{pol} \;=\; -\frac{1}{2}\Big(1-\frac{1}{\varepsilon}\Big)\sum_{kk'} q_k q_{k'} \gamma_{kk'}, \qquad \gamma_{kk'} \;=\; \Big[R_{kk'}^2 + \alpha_k \alpha_{k'} e^{-R_{kk'}^2/d\,\alpha_k\alpha_{k'}}\Big]^{-1/2}, \tag{58}$$

with Still's interpolating kernel and $\gamma_{kk} = 1/\alpha_k$. Since the charges are linear in the density, the SCF minimizes $E + \Delta G_{pol}$ directly rather than through an outer iteration, and the energy it prints is the free energy above.

The effective radii α_k come either from the pairwise descreening expression, which is closed form and cheap, or from the Coulomb-field integral, which is what the SMx models were fitted with; `BORN_RADII` chooses. Analytic gradients are available for both, and for the CM5, Mulliken and Löwdin charges, with any of the five references.

22.9.1 GB Module Options

- **SASA_TYPE**
  TYPE: STRING
  DEFAULT: GEPOL
  OPTIONS: GEPOL, NUMERIC, LCPO
  DESCRIPTION: how the solvent-accessible surface area of each atom is computed. GEPOL discretizes each sphere and drops the buried tesserae; NUMERIC uses a cylindrical grid; LCPO is an analytic pairwise approximation.

- **CHARGE_TYPE**
  TYPE: STRING
  DEFAULT: CM5
  OPTIONS: CM5, IAO_MULLIKEN, IAO_LOWDIN, MULLIKEN, LOWDIN
  DESCRIPTION: which charges drive the electrostatics. CM5 is the class-IV model above, is the least basis-dependent, and is the only one of the five whose derivative is implemented. Mulliken and Löwdin also have gradients. The two intrinsic-orbital variants need a closed-shell reference and are energy only: they are not linear in the density, because the IAO coefficients are themselves built from the occupied orbitals, so the Fock contribution is incomplete and $E + \Delta G_{pol}$ is not stationary. `MacroQC` prints a warning when they are selected.

- **CDS_TYPE**
  TYPE: STRING
  DEFAULT: SMD
  OPTIONS: SMD, SM8AD, SM12
  DESCRIPTION: the parameterization of the non-electrostatic term. SMD and SM8AD differ only in the coefficients of the molecular term; SM12 is the larger set of surface tensions described in Section 22.10.

- **CDS_GRADIENT**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: print the analytic derivative of the non-electrostatic term. It also switches the surface areas, which is what makes them differentiable and which moves G_{CDS} by a few thousandths of a kcal/mol; the electrostatic half has no derivative, so this is not a total gradient.

- **COULOMB_RADII**
  TYPE: STRING
  DEFAULT: SMD
  OPTIONS: SMD, SM8, SM12
  DESCRIPTION: the intrinsic Coulomb radii the Born radii are built from. The SM12 set is the only one whose radii follow the solvent.

- **BORN_RADII**
  TYPE: STRING
  DEFAULT: ANALYTIC
  OPTIONS: ANALYTIC, CFA
  DESCRIPTION: how the effective Born radii are obtained. ANALYTIC is the pairwise descreening expression, which is fast but counts the overlap of two neighbours twice; CFA evaluates the Coulomb-field integral directly and is what the SMx models use.

- **BORN_LEBEDEV**
  TYPE: INTEGER
  DEFAULT: 41
  DESCRIPTION: the Lebedev order used for the solid angle in the CFA integral.

- **BORN_SWITCH**
  TYPE: DOUBLE
  DEFAULT: 0.15
  DESCRIPTION: half-width, in bohr, of the switching function that decides whether a point of the Coulomb-field integral lies inside a neighbour's sphere. With a sharp cutoff a point flips from buried to exposed all at once, the effective radii are step functions of the geometry, and there is no derivative; a positive width is required for a CFA gradient. The bias is a few thousandths of a kcal/mol and is settled by 0.15.

- **GRADIENT**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: switch the surfaces that G_{CDS} is built on even for an energy job, so that the energy matches the one a derivative job reports. Any derivative jobtype does this on its own.

- **RADIUS_TYPE**
  TYPE: STRING
  DEFAULT: SMD
  OPTIONS: SMD, ALVAREZ, COV_PLUS_CONST, BONDI
  DESCRIPTION: the van der Waals radii the surface area is computed on.

- **SOLVENT_PARAMS**
  TYPE: STRING
  DEFAULT: SM8
  OPTIONS: SM8, MN21
  DESCRIPTION: which set of solvent descriptors to use.

- **COVALENT_VW_CONST**
  TYPE: DOUBLE
  DEFAULT: 0.76
  DESCRIPTION: the constant added to a covalent radius when RADIUS_TYPE is COV_PLUS_CONST.

- **gb_dij**, **gb_sij**, **gb_cij**
  TYPE: DOUBLE
  DEFAULT: 3.7, 0.75, 0.0
  DESCRIPTION: the empirical constants of the generalized-Born interaction kernel and of the descreening expression.

- **SOLVENT**
  TYPE: STRING
  DEFAULT: WATER
  DESCRIPTION: the solvent, from the list of Table **??**.

22.10 SM12

SM12 [102] is the generalized-Born model above with three things fixed: the charges are CM5, the intrinsic Coulomb radii are its own, and the non-electrostatic term uses the considerably larger set of atomic surface tensions introduced with SM6 [103], which is richer than SMD's. Requesting it,

```
$set_scf
  solvation true
  solvation_model sm12
$end_scf

$set_gb
  solvent water
$end_gb
```

sets `CHARGE_TYPE CM5`, `COULOMB_RADII SM12`, `CDS_TYPE SM12`, `BORN_RADII CFA` and the SM8 tables of solvent descriptors; any of them may still be overridden in the `GB` block.

The electrostatic half is the generalized-Born expression

$$\Delta G_{elec} \;=\; -\frac{1}{2}\Big(1-\frac{1}{\varepsilon}\Big)\sum_{kk'} q_k q_{k'} \gamma_{kk'}, \qquad \gamma_{kk'} = \Big(R_{kk'}^2 + \alpha_k \alpha_{k'} e^{-R_{kk'}^2/d\,\alpha_k\alpha_{k'}}\Big)^{-1/2}, \quad (59)$$

with $d = 3.7$, the CM5 charges q_k of Section 22 answering the density self-consistently, and the effective Born radii α_k from the Coulomb-field integral. Seven of the intrinsic Coulomb radii [104] the α_k are built from — H, O, F, P, S, Cl and Br — follow the solvent through Abraham's hydrogen bond acidity,

$$\rho_Z \;=\; \rho_Z^{aq} \;+\; A_Z\,(\alpha - 0.43), \qquad \alpha < 0.43, \quad (60)$$

and keep their aqueous value above that.

The non-electrostatic half has the same shape as in SMD,

$$G_{CDS} \;=\; \sum_k A_k \sigma_k \;+\; \sigma_M \sum_k A_k, \tag{61}$$

on the same surface — van der Waals spheres grown by a 0.4 Å probe — but the atomic surface tension σ_k carries nineteen pair and functional group terms instead of ten. Each of them is switched on by

$$T(R_{kk'}) \;=\; \begin{cases} \exp\!\Big[\dfrac{\Delta R}{R_{kk'} - R_{ZZ'} - \Delta R}\Big], & R_{kk'} < R_{ZZ'} + \Delta R \\ 0, & \text{otherwise} \end{cases} \tag{62}$$

with a distance $R_{ZZ'}$ that depends on the two elements. The terms that make SM12 different from SMD are the ones that reach past a single bond: the amide and urea nitrogen, which counts the oxygens carried by the carbon it sits on,

$$\sigma_N \;\mathrel{+}=\; \sigma^{N\text{–}C}\Big[\sum_C T(R_{NC})\Big(\sum_{m\neq N,C} T(R_{Cm})\Big)^2\Big]^{1.3} \;+\; \sigma^{N\text{–}C(2)} \sum_C T(R_{NC}) \sum_{O\neq N,C} T(R_{CO}), \tag{63}$$

the nitrile nitrogen and the doubly bonded carbon, which are given their own short cut-off distances, and the carboxyl carbon, which counts ordered pairs of oxygens,

$$\sigma_C \;\mathrel{+}=\; \sigma^{C\text{–}O(2)} \sum_O \sum_{O'\neq O} T(R_{CO})\, T(R_{CO'}). \tag{64}$$

For water the σ were fitted directly and σ_M vanishes; for any other solvent each of them is a linear combination of the refractive index and Abraham's acidity and basicity, and

$$\sigma_M \;=\; 0.17\,\gamma \;+\; 6.98\,\beta^2 \;-\; 2.30\,\phi^2 \;-\; 7.02\,\psi^2. \tag{65}$$

The derivative of G_{CDS} is analytic, and so is the derivative of the electrostatic half, so SM12 has a full analytic gradient. On water in water it agrees with central differences of the `MacroQC` energy to 1.6×10^{-7} hartree/bohr, and with central differences of an independent implementation's free energy to 1.5×10^{-5}, the residual being the 2.8% difference in the Coulomb-field solid angle noted above.

SM12 refuses to run on an element it was never fitted for rather than substituting a zero surface tension: the published tables carry a sentinel there, and a silent answer would be wrong by tens of kcal/mol.

22.11 VASPsol

The self-consistent continuum model of Fattebert and Gygi [105], in the form of the VASPsol implementation [106, 107], has no cavity at all. The permittivity is a smooth function of the solute's own electron density,

$$\varepsilon(\mathbf{r}) \;=\; 1 \;+\; (\varepsilon_b - 1)\, S(\mathbf{r}), \qquad S \;=\; \frac{1}{2}\operatorname{erfc}\!\left[\frac{\ln(\rho/n_c)}{\sqrt{2}\,\sigma}\right], \tag{66}$$

so it is one where the density is high and ε_b where the density has died away, and the boundary between them moves as the density polarizes. There are no radii: the only parameters are the critical density n_c, the width σ and a surface tension τ, three numbers for the whole periodic table.

What has to be solved is the generalized Poisson equation, $\nabla \cdot (\varepsilon \nabla \phi) = -4\pi \rho_{tot}$, which has no Green's function and is solved on a real-space grid by preconditioned conjugate gradients. The non-electrostatic term is a single surface tension on the quantum surface area,

$$G_{non-elec} \;=\; \tau \int |\nabla S(\mathbf{r})| \, d^3 r, \tag{67}$$

which needs neither radii nor a tessellation.

Because the cavity is made of the density, the potential the SCF sees carries two terms that no apparent-charge model has: the dielectric follows the density, and so does the surface. Both are included.

The grid error is a clean h^2, and its size is set by the transition shell, which is about as thick as a reasonable spacing. Rather than force a very fine grid, the model repeats the solvation energy once at the converged density on a finer grid and extrapolates to $h \to 0$; this is on by default and costs one extra Poisson solve. Turning it off and tightening `SPACING` by hand gives the same answer more slowly.

Analytic gradients are available. The SCF minimizes the energy including the solvation terms, so no orbital response is needed; what is left is the derivative of the two solvation terms at fixed density, which is the nuclei moving in the reaction field plus the basis functions moving in the solvation potential. On water in water the total gradient agrees with central differences of the total energy to 2×10^{-6} E_h/bohr.

The parameters are those published for water and were fitted against pseudopotential valence densities. They have not been refitted for all-electron Gaussian densities, so the model should be treated as implemented rather than as calibrated.

22.11.1 VASPSOL Module Options

- **SOLVENT**
  TYPE: STRING
  DEFAULT: WATER
  DESCRIPTION: names the bulk permittivity in the solvent table at the end of this section.

- **EPSILON**
  TYPE: DOUBLE
  DEFAULT: -1
  DESCRIPTION: a positive value overrides the solvent table.

- **CRITICAL_DENSITY**
  TYPE: DOUBLE

DEFAULT: 0.0025
DESCRIPTION: n_c, the density at which the medium is half solvent, in Å^{-3}.

- **SIGMA**
  TYPE: DOUBLE
  DEFAULT: 0.6
  DESCRIPTION: the width of the transition, in $\ln \rho$; dimensionless.

- **TAU**
  TYPE: DOUBLE
  DEFAULT: 0.525
  DESCRIPTION: the effective surface tension of the quantum surface area, in $\text{meV}/\text{Å}^2$.

- **SPACING**, **PADDING**
  TYPE: DOUBLE
  DEFAULT: 0.15 and 4.0
  DESCRIPTION: the cubic grid the Poisson equation is solved on, in Å, and how far it extends beyond the outermost nucleus. The padding only has to be large enough that the solute density has died away at the box faces.

- **EXTRAPOLATE**
  TYPE: BOOL
  DEFAULT: TRUE
  DESCRIPTION: repeat the solvation energy once on a finer grid at the converged density and extrapolate in h^2.

- **EXTRAPOLATE_RATIO**
  TYPE: DOUBLE
  DEFAULT: 0.75
  DESCRIPTION: how much finer the second grid is. Smaller is more accurate and more expensive.

- **SOLVER_MAXITER**, **SOLVER_CONVERGENCE**
  TYPE: INTEGER, DOUBLE
  DEFAULT: 2000 and 10^{-9}
  DESCRIPTION: the iteration cap and the relative residual of the conjugate-gradient solver.

- **GAS_PHASE**
  TYPE: BOOL
  DEFAULT: FALSE
  DESCRIPTION: also run the solute in vacuum, so that the solvation free energy can be reported against it rather than only as the sum of its solvated halves.

- **KAPPA2**
  TYPE: DOUBLE
  DEFAULT: 0.0

DESCRIPTION: Debye screening of a dilute electrolyte, in bohr^{-2}. Read but not yet used.

22.12 What Is Available for Each Model

Every model runs with the RHF, UHF, ROHF, RKS and UKS references. What differs is whether a nuclear gradient exists, and therefore whether the model can be used for anything beyond a single point.

model	energy	gradient	notes
`GB`	yes	yes	needs a differentiable charge model: CM5, Mulliken or Löwdin. The intrinsic-orbital charges are energy only.
`SM12`	yes	yes	electrostatics and the non-electrostatic term both.
`COSMO`	yes	yes	RHF, UHF and DFT. Not with a correlated density.
`CPCM`, `IEFPCM`, `GCOSMO`	yes	yes	needs the switched surface, which `PCM GRADIENT TRUE` turns on and every gradient jobtype turns on by itself. Available with the Gaussian kernel as well.
`VASPSOL`	yes	yes	extrapolated in h^2 alongside the energy.
`SMD`	yes	yes	`JOBTYPE ENERGY` runs the solute in vacuum and in the continuum and reports ΔG_{solv}. Every derivative jobtype — `GRAD`, `OPT`, `FREQ`, `IRC` — skips the vacuum run instead and works with the solvated free energy $A = E + G_{pol} + G_{CDS}$ and its derivative, which is what moving the nuclei needs.
`CMIRS`	yes	numerical	the four non-electrostatic terms are functionals of the density and the continuum runs on a cavity the density itself defines, so neither the PCM nor the surface-tension derivative applies. `GRAD`, `OPT` and `FREQ` difference the solvated free energy $A = E_{sol} + G_{nonel}$ instead, on the ray form of the non-electrostatic terms; see below.
`COSMORS`	yes	no	a post-processing step on a COSMO surface.

Every gradient in the table is checked against central differences of the energy it belongs to, which is the check that cannot be fooled by a convention: COSMO to 10^{-7} E_h/bohr, the generalized-Born and SM12 gradients to 10^{-7}, C-PCM and IEF-PCM to 2×10^{-6}, VASPsol to 10^{-6}. Where another program offers the same model the two are compared directly as well; the COSMO gradient agrees with Molpro to 10^{-7} E_h/bohr over a set of nine molecules at Hartree–Fock.

The CMIRS gradient is the exception and is worth stating plainly. It is numerical, and what it is worth depends on which of the model's two ways of evaluating the non-electrostatic

terms is used. NONELS_GRID CUBE, the cubic grid the published numbers were produced with, decides which region a point belongs to by a test on the density there, so a point that crosses the contour as the nuclei move takes a whole cell with it and the energy is a staircase. NONELS_GRID RAY writes the same four terms on the rays of the isodensity surface, where the boundary is resolved exactly: the exchange integral by Gauss–Laguerre along each ray, the dispersion integral in closed form as a sum over the solid-angle elements. The two agree on the energy to 0.03 kcal/mol, which is inside the cubic grid's own discretization error, and the rays are the faster of the two.

They do not agree on how well they can be differenced. The sum of the forces, which a translation-invariant energy makes vanish and which is therefore a direct measure of the error, is 1.5×10^{-2} E_h/bohr on the cubic grid at its default spacing, 1.2×10^{-4} with the spacing halved, and 9×10^{-6} on the rays; the out-of-plane force on a planar solute, zero by symmetry, is 4×10^{-3}, 1×10^{-3} and zero to 10^{-10}. A derivative jobtype therefore selects RAY unless the user has asked otherwise. Asked for CUBE anyway, it freezes the grid across the displacements, halves CMIRS SPACING to 0.125 Å and relaxes OPT TOL_MAXG and TOL_RMSG to 3×10^{-3} and 2×10^{-3}, because a force built that way cannot meet the defaults. The displacement is STEP_BOHR, whose default of 0.005 is too small here; 0.02 is the right order.

22.13 Available Solvents

The solvent descriptors every model here needs are tabulated for the following solvents, which are what the SOLVENT option of every solvation block accepts. Names carry no spaces, hyphens or commas: 1,1,1-trichloroethane is written 111TRICHLOROETHANE. A solvent that is missing any one of its six descriptors stops the run rather than answering with a zero, so argon, krypton and xenon — which have dielectric constants but were never given descriptors — are available to the purely electrostatic models and refused by SMD and SM12.

Table 9: Available Solvent List in MacroQC.

111TRICHLOROETHANE	112TRICHLOROETHANE	124TRIMETHYLBENZENE
12ETHANEDIOL	14DIOXANE	1BROMO2METHYLPROPANE
1BROMOOCTANE	1BROMOPENTANE	1BROMOPROPANE
1BUTANOL	1CHLOROHEXANE	1CHLOROPENTANE
1CHLOROPROPANE	1DECANOL	1FLUOROOCTANE
1HEPTANOL	1HEXANOL	1HEXENE
1HEXYNE	1IODOBUTANE	1IODOHEXADECANE
1IODOPENTANE	1IODOPROPANE	1NITROPROPANE
1NONANOL	1OCTANOL	1PENTANOL
1PENTENE	1PROPANOL	222TRIFLUOROETHANOL
224TRIMETHYLPENTANE	24DIMETHYLPENTANE	24DIMETHYLPYRIDINE
26DIMETHYLPYRIDINE	2BROMOPROPANE	2BUTANOL
2CHLOROBUTANE	2HEPTANONE	2HEXANONE
2METHOXYETHANOL	2METHYL1PROPANOL	2METHYL2PROPANOL
2METHYLPENTANE	2METHYLPYRIDINE	2NITROPROPANE
2OCTANONE	2PENTANONE	2PROPANOL
2PROPEN1OL	3METHYLPYRIDINE	3PENTANONE

4HEPTANONE	4METHYL2PENTANONE	4METHYLPYRIDINE
5NONANONE	ACETICACID	ACETONE
ACETONITRILE	ACETOPHENONE	ACHLOROTOLUENE
ANILINE	ANISOLE	BENZALDEHYDE
BENZENE	BENZONITRILE	BENZYLALCOHOL
BROMOBENZENE	BROMOETHANE	BROMOFORM
BUTANAL	BUTANOICACID	BUTANONE
BUTANONITRILE	BUTYLAMINE	BUTYLETHANOATE
CARBONDISULFIDE	CARBONTETRACHLORIDE	CHLOROBENZENE
CHLOROFORM	CIS12DIMETHYLCYCLOHEXANE	CISDECALIN
CYCLOHEXANE	CYCLOHEXANONE	CYCLOPENTANE
CYCLOPENTANOL	CYCLOPENTANONE	DECALINCISTRANSMIXTURE
DIBROMOETHANE	DIBROMOMETHANE	DIBUTYLETHER
DICHLOROETHANE	DICHLOROMETHANE	DIETHYLAMINE
DIETHYLETHER	DIETHYLSULFIDE	DIIODOMETHANE
DIISOPROPYLETHER	DIMETHYLDISULFIDE	DIMETHYLSULFOXIDE
DIPHENYLETHER	DIPROPYLAMINE	E12DICHLOROETHENE
E2PENTENE	ETHANETHIOL	ETHANOL
ETHYLBENZENE	ETHYLETHANOATE	ETHYLMETHANOATE
ETHYLPHENYLETHER	FLUOROBENZENE	FORMAMIDE
FORMICACID	HEPTANE	HEXANOICACID
IODOBENZENE	IODOETHANE	IODOMETHANE
ISOPROPYLBENZENE	ISOQUINOLINE	MCRESOL
MESITYLENE	METHANOL	METHYLBENZOATE
METHYLBUTANOATE	METHYLCYCLOHEXANE	METHYLETHANOATE
METHYLMETHANOATE	METHYLPROPANOATE	MXYLENE
NBUTYLBENZENE	NDECANE	NDODECANE
NHEPTANE	NHEXADECANE	NHEXANE
NITROBENZENE	NITROETHANE	NITROMETHANE
NMETHYLANILINE	NMETHYLFORMAMIDE	NMETHYLFORMAMIDEEZMIXTURE
NNDIMETHYLACETAMIDE	NNDIMETHYLFORMAMIDE	NNONANE
NOCTANE	NPENTADECANE	NPENTANE
NUNDECANE	OCHLOROTOLUENE	OCRESOL
ODICHLOROBENZENE	ONITROTOLUENE	OXYLENE
PENTADECANE	PENTANAL	PENTANOICACID
PENTYLAMINE	PENTYLETHANOATE	PERFLUOROBENZENE
PISOPROPYLTOLUENE	PROPANAL	PROPANOICACID
PROPANONITRILE	PROPYLAMINE	PROPYLETHANOATE
PXYLENE	PYRIDINE	QUINOLINE
SECBUTYLBENZENE	TERTBUTYLBENZENE	TETRACHLOROETHENE
TETRAHYDROFURAN	TETRAHYDROTHIOPHENESSDIOXIDE	TETRALIN
THIOPHENE	THIOPHENOL	TOLUENE
TRANSDECALIN	TRIBUTYLPHOSPHATE	TRICHLOROETHENE
TRIETHYLAMINE	WATER	XYLENE
XYLENEMIXTURE	Z12DICHLOROETHENE	

References

[1] J. L. Whitten, J. Chem. Phys. **58**, 4496 (1973).

[2] B. I. Dunlap, J. W. D. Connolly, and J. R. Sabin, J. Chem. Phys. **71**, 3396 (1979).

[3] M. Feyereisen, G. Fitzgerald, and A. Komornicki, Chem. Phys. Lett. **208**, 359 (1993).

[4] O. Vahtras, J. Almlöf, and M. W. Feyereisen, Chem. Phys. Lett. **213**, 514 (1993).

[5] A. P. Rendell and T. J. Lee, J. Chem. Phys. **101**, 400 (1994).

[6] F. Weigend, Phys. Chem. Chem. Phys. **4**, 4285 (2002).

[7] A. Sodt, J. E. Subotnik, and M. Head-Gordon, J. Chem. Phys. **125**, 194109 (2006).

[8] H.-J. Werner and M. Schütz, J. Chem. Phys. **135**, 144116 (2011).

[9] A. E. DePrince and C. D. Sherrill, J. Chem. Theory Comput. **9**, 2687 (2013).

[10] U. Bozkaya, J. Chem. Theory Comput. **10**, 2371 (2014).

[11] U. Bozkaya, J. Chem. Phys. **141**, 124108 (2014).

[12] U. Bozkaya, J. Chem. Theory Comput. **10**, 4389 (2014).

[13] U. Bozkaya, J. Chem. Theory Comput. **12**, 1179 (2016).

[14] U. Bozkaya, Phys. Chem. Chem. Phys. **18**, 11362 (2016).

[15] U. Bozkaya, J. Chem. Phys. **144**, 144108 (2016).

[16] U. Bozkaya and C. D. Sherrill, J. Chem. Phys. **144**, 174103 (2016).

[17] U. Bozkaya and C. D. Sherrill, J. Chem. Phys. **147**, 044104 (2017).

[18] U. Bozkaya, J. Comput. Chem. **39**, 351 (2018).

[19] U. Bozkaya, J. Chem. Theory Comput. **15**, 4415 (2019).

[20] U. Bozkaya, Int. J. Quant. Chem. **121**, e26623 (2021).

[21] T. H. Dunning, J. Chem. Phys. **90**, 1007 (1989).

[22] D. E. Woon and T. H. Dunning, J. Chem. Phys. **103**, 4572 (1995).

[23] T. Helgaker, P. Jørgensen, and J. Olsen, *Molecular Electronic Structure Theory*, 1 ed. (John Wiley & Sons, New York, 2000), pp. 336–426.

[24] H. B. Schlegel and M. J. Frisch, Int. J. Quant. Chem. **54**, 83 (1995).

[25] F. Weigend, A. Köhn, and C. Hättig, J. Chem. Phys. **116**, 3175 (2002).

[26] B. P. Pritchard, D. Altarawy, B. Didier, T. D. Gibson, and T. L. Windus, J. Chem. Inf. Model. **59**, 4814 (2019).

[27] G. Chaban, M. W. Schmidt, and M. S. Gordon, Theor. Chem. Acc. **97**, 88 (1997).

[28] M. Wolfsberg and L. Helmholtz, J. Chem. Phys. **20**, 837 (1952).

[29] M. Guest and V. R. Saunders, Mol. Phys. **28**, 819 (1974).

[30] C. C. J. Roothaan, Rev. Mod. Phys. **32**, 179 (1960).

[31] E. R. Davidson, Chem. Phys. Lett. **21**, 565 (1973).

[32] J. Binkley, J. Pople, and P. Dobosh, Mol. Phys. **28**, 1423 (1974).

[33] R. McWeeny and G. Diercksen, J. Chem. Phys. **49**, 4852 (1968).

[34] K. Fægri and R. Manne, Mol. Phys. **31**, 1037 (1976).

[35] B. N. Plakhutin, E. V. Gorelik, and N. N. Breslavskaya, J. Chem. Phys. **125**, 204110 (2006).

[36] G. M. J. Barca, C. Bertoni, L. Carrington, D. Datta, N. D. Silva, J. E. Deustua, D. G. Fedorov, J. R. Gour, A. O. Gunina, E. Guidez, T. Harville, S. Irle, J. Ivanic, K. Kowalski, S. S. Leang, H. Li, W. Li, J. J. Lutz, I. Magoulas, J. Mato, V. Mironov, H. Nakata, B. Q. Pham, P. Piecuch, D. Poole, S. R. Pruitt, A. P. Rendell, L. B. Roskop, K. Ruedenberg, T. Sattasathuchana, M. W. Schmidt, J. Shen, L. Slipchenko, M. Sosonkina, V. Sundriyal, A. Tiwari, J. L. G. Vallejo, B. Westheimer, M. Włoch, P. Xu, F. Zahariev, and M. S. Gordon, J. Chem. Phys. **152**, 154102 (2020).

[37] H. B. Schlegel, WIREs Computational Molecular Science **1**, 790 (2011).

[38] F. Jensen, *Introduction to computational chemistry* (John wiley & sons, Chichester, 2017), pp. 316–338.

[39] W. H. Miller, N. C. Handy, and J. E. Adams, J. Chem. Phys. **72**, 99 (1980).

[40] A. Behn, P. M. Zimmerman, A. T. Bell, and M. Head-Gordon, The Journal of chemical physics **135**, 224108 (2011).

[41] T. A. Halgren and W. N. Lipscomb, Chem. Phys. Lett. **49**, 225 (1977).

[42] C. Peng and H. B. Schlegel, Israel Journal of Chemistry **33**, 449 (1993).

[43] P. M. Zimmerman, The Journal of Chemical Physics **138**, 184102 (2013).

[44] S. M. Sharada, P. M. Zimmerman, A. T. Bell, and M. Head-Gordon, Journal of chemical theory and computation **8**, 5166 (2012).

[45] A. Goodrow, A. T. Bell, and M. Head-Gordon, The Journal of Chemical Physics **129**, 174109 (2008).

[46] J. Baker, A. Kessi, and B. Delley, The Journal of Chemical Physics **105**, 192 (1996).

[47] G. Henkelman and H. Jónsson, The Journal of chemical physics **113**, 9978 (2000).

[48] G. Henkelman, B. P. Uberuaga, and H. Jónsson, The Journal of chemical physics **113**, 9901 (2000).

[49] P. M. Zimmerman, Journal of Chemical Theory and Computation **9**, 3043 (2013).

[50] P. M. Zimmerman, Journal of Computational Chemistry **36**, 601 (2015).

[51] G. Mills and H. Jónsson, Physical review letters **72**, 1124 (1994).

[52] B. Ermis, A. Unal, E. Soydas, and U. Bozkaya (unpublished).

[53] D. G. A. Smith, L. A. Burns, A. C. Simmonett, R. M. Parrish, M. C. Schieber, R. Galvelis, P. Kraus, H. Kruse, R. D. Remigio, A. Alenaizan, A. M. James, S. Lehtola, J. P. Misiewicz, M. Scheurer, R. A. Shaw, J. B. Schriber, Y. Xie, Z. L. Glick, D. A. Sirianni, J. S. O'Brien, J. M. Waldrop, A. Kumar, E. G. Hohenstein, B. P. Pritchard, B. R. Brooks, H. F. Schaefer, A. Y. Sokolov, K. Patkowski, A. E. DePrince, U. Bozkaya, R. A. King, F. A. Evangelista, J. M. Turney, T. D. Crawford, and C. D. Sherrill, J. Chem. Phys. **152**, 184108 (2020).

[54] Y. Shao *et al.*, Molecular Physics **113**, 184 (2014).

[55] D. A. Matthews, L. Cheng, M. E. Harding, F. Lipparini, S. Stopkowicz, T.-C. Jagau, P. G. Szalay, J. Gauss, and J. F. Stanton, J. Chem. Phys. **152**, 214108 (2020).

[56] H.-J. Werner, P. J. Knowles, F. R. Manby, J. A. Black, K. Doll, A. Heßelmann, D. Kats, A. Köhn, T. Korona, D. A. Kreplin, Q. Ma, T. F. Miller, A. Mitrushchenkov, K. A. Peterson, I. Polyak, G. Rauhut, and M. Sibaev, J. Chem. Phys. **152**, 144107 (2020).

[57] F. Neese, WIREs Comput. Mol. Sci. **2**, 73 (2012).

[58] D. A. Clabo, W. D. Allen, R. B. Remington, Y. Yamaguchi, and H. F. Schaefer, Chem. Phys. **123**, 187 (1988).

[59] J. M. L. Martin, T. J. Lee, P. R. Taylor, and J.-P. François, J. Chem. Phys. **103**, 2589 (1995).

[60] U. Bozkaya, J. M. Turney, Y. Yamaguchi, H. F. Schaefer, and C. D. Sherrill, J. Chem. Phys. **135**, 104103 (2011).

[61] U. Bozkaya and C. D. Sherrill, J. Chem. Phys. **138**, 184103 (2013).

[62] U. Bozkaya, J. Chem. Phys. **135**, 224103 (2011).

[63] E. Soydaş and U. Bozkaya, J. Chem. Theory Comput. **9**, 1452 (2013).

[64] U. Bozkaya, J. Chem. Phys. **139**, 104116 (2013).

[65] U. Bozkaya and C. D. Sherrill, J. Chem. Phys. **141**, 204105 (2014).

[66] E. Soydaş and U. Bozkaya, J. Chem. Theory Comput. **11**, 1564 (2015).

[67] U. Bozkaya and C. D. Sherrill, J. Chem. Phys. **139**, 054104 (2013).

[68] E. Soydaş and U. Bozkaya, J. Comput. Chem. **35**, 1073 (2014).

[69] U. Bozkaya, A. Ünal, and Y. Alagöz, J. Chem. Phys. **153**, 244115 (2020).

[70] U. Bozkaya and H. F. Schaefer, J. Chem. Phys. **136**, 204114 (2012).

[71] E. Epifanovsky, D. Zuev, X. Feng, K. Khistyaev, Y. Shao, and A. I. Krylov, J. Chem. Phys. **139**, 134105 (2013).

[72] A. Ünal and U. Bozkaya, J. Chem. Theory Comput. **18**, 1489 (2022).

[73] A. Ünal and U. Bozkaya, J. Comput. Chem. **45**, 2969 (2024).

[74] A. Ünal and U. Bozkaya, J. Chem. Phys. **161**, 114108 (2024).

[75] Y. Shao, L. Fusti-Molnar, Y. Jung, J. Kussmann, C. Ochsenfeld, S. T. Brown, A. T. B. Gilbert, L. V. Slipchenko, S. V. Levchenko, D. P. O'Neill, R. A. DiStasio, R. C. Lochan, T. Wang, G. J. O. Beran, N. A. Besley, J. M. Herbert, C. Y. Lin, T. V. Voorhis, S. H. Chien, A. Sodt, R. P. Steele, V. A. Rassolov, P. E. Maslen, P. P. Korambath, R. D. Adamson, B. Austin, J. Baker, E. F. C. Byrd, H. Dachsel, R. J. Doerksen, A. Dreuw, B. D. Dunietz, A. D. Dutoi, T. R. Furlani, S. R. Gwaltney, A. Heyden, S. Hirata, C. P. Hsu, G. Kedziora, R. Z. Khaliullin, P. Klunzinger, A. M. Lee, M. S. Lee, W. Liang, I. Lotan, N. Nair, B. Peters, E. I. Proynov, P. A. Pieniazek, Y. M. Rhee, J. Ritchie, E. Rosta, C. D. Sherrill, A. C. Simmonett, J. E. Subotnik, H. L. Woodcock, W. Zhang, A. T. Bell, A. K. Chakraborty, D. M. Chipman, F. J. Keil, A. Warshel, W. J. Hehre, H. F. Schaefer, J. Kong, A. I. Krylov, P. M. W. Gill, and M. Head-Gordon, Phys. Chem. Chem. Phys. **8**, 3172 (2006).

[76] P. Valiron and I. Mayer, Chem. Phys. Lett. **275**, 46 (1997).

[77] D. W. Zhang and J. Zhang, The Journal of chemical physics **119**, 3599 (2003).

[78] J. Tomasi, B. Mennucci, and R. Cammi, Chem. Rev. **105**, 2999 (2005).

[79] J. L. Pascual-Ahuir, E. Silla, and I. T. nón, J. Comput. Chem. **15**, 1127 (1994).

[80] D. M. York and M. Karplus, J. Phys. Chem. A **103**, 11060 (1999).

[81] A. W. Lange and J. M. Herbert, J. Chem. Phys. **133**, 244111 (2010).

[82] A. Bondi, J. Phys. Chem. **68**, 441 (1964).

[83] M. Mantina, A. C. Chamberlin, R. Valero, C. J. Cramer, and D. G. Truhlar, J. Phys. Chem. A **113**, 5806 (2009).

[84] A. K. Rappé, C. J. Casewit, K. S. Colwell, W. A. G. III, and W. M. Skiff, J. Am. Chem. Soc. **114**, 10024 (1992).

[85] A. Klamt and G. Schüürmann, J. Chem. Soc., Perkin Trans. 2 **1993**, 799 (1993).

[86] S. Miertuš, E. Scrocco, and J. Tomasi, Chem. Phys. **55**, 117 (1981).

[87] V. Barone and M. Cossi, J. Phys. Chem. A **102**, 1995 (1998).

[88] T. N. Truong and E. V. Stefanovich, Chem. Phys. Lett. **240**, 253 (1995).

[89] E. Cancès, B. Mennucci, and J. Tomasi, J. Chem. Phys. **107**, 3032 (1997).

[90] D. M. Chipman, J. Chem. Phys. **112**, 5558 (2000).

[91] R. A. Pierotti, Chem. Rev. **76**, 717 (1976).

[92] A. V. Marenich, C. J. Cramer, and D. G. Truhlar, J. Phys. Chem. B **113**, 6378 (2009).

[93] A. Pomogaeva and D. M. Chipman, J. Chem. Theory Comput. **7**, 3952 (2011).

[94] A. Pomogaeva and D. M. Chipman, J. Chem. Theory Comput. **10**, 211 (2014).

[95] Z.-Q. You and J. M. Herbert, J. Chem. Theory Comput. **12**, 4638 (2016).

[96] O. A. Vydrov and T. V. Voorhis, J. Chem. Phys. **130**, 104105 (2009).

[97] A. Klamt, J. Phys. Chem. **99**, 2224 (1995).

[98] A. Klamt, V. Jonas, T. Bürger, and J. C. W. Lohrenz, J. Phys. Chem. A **102**, 5074 (1998).

[99] F. L. Hirshfeld, Theor. Chim. Acta **44**, 129 (1977).

[100] A. V. Marenich, S. V. Jerome, C. J. Cramer, and D. G. Truhlar, J. Chem. Theory Comput. **8**, 527 (2012).

[101] W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson, J. Am. Chem. Soc. **112**, 6127 (1990).

[102] A. V. Marenich, C. J. Cramer, and D. G. Truhlar, J. Chem. Theory Comput. **9**, 609 (2013).

[103] C. P. Kelly, C. J. Cramer, and D. G. Truhlar, J. Chem. Theory Comput. **1**, 1133 (2005).

[104] A. V. Marenich, R. M. Olson, C. P. Kelly, C. J. Cramer, and D. G. Truhlar, J. Chem. Theory Comput. **3**, 2011 (2007).

[105] J.-L. Fattebert and F. Gygi, J. Comput. Chem. **23**, 662 (2002).

[106] K. Mathew, R. Sundararaman, K. Letchworth-Weaver, T. A. Arias, and R. G. Hennig, J. Chem. Phys. **140**, 084106 (2014).

[107] K. Mathew, V. S. C. Kolluru, S. Mula, S. N. Steinmann, and R. G. Hennig, J. Chem. Phys. **151**, 234101 (2019).